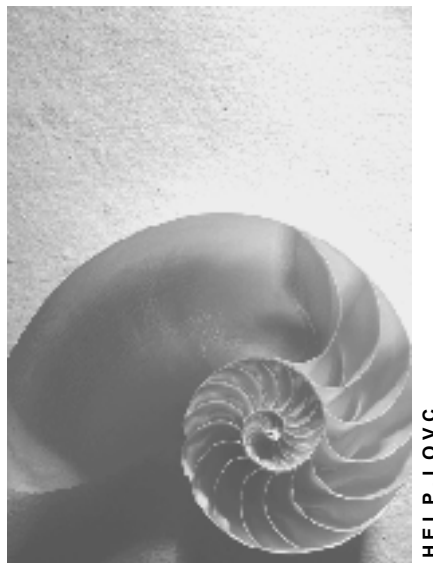


# Variant Configuration (LO-VC)



**Release 4.6C**



## Copyright

© Copyright 2000 SAP AG. All rights reserved.

No part of this brochure may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft®, WINDOWS®, NT®, EXCEL®, Word® and SQL Server® are registered trademarks of Microsoft Corporation.

IBM®, DB2®, OS/2®, DB2/6000®, Parallel Sysplex®, MVS/ESA®, RS/6000®, AIX®, S/390®, AS/400®, OS/390®, and OS/400® are registered trademarks of IBM Corporation.

ORACLE® is a registered trademark of ORACLE Corporation, California, USA.

INFORMIX®-OnLine for SAP and Informix® Dynamic Server™ are registered trademarks of Informix Software Incorporated.

UNIX®, X/Open®, OSF/1®, and Motif® are registered trademarks of The Open Group.







HTML, DHTML, XML, XHTML are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Laboratory for Computer Science NE43-358, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139.

JAVA® is a registered trademark of Sun Microsystems, Inc. , 901 San Antonio Road, Palo Alto, CA 94303 USA.

JAVASCRIPT® is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, SAP Logo, mySAP.com, mySAP.com Marketplace, mySAP.com Workplace, mySAP.com Business Scenarios, mySAP.com Application Hosting, WebFlow, R/2, R/3, RIVA, ABAP, SAP Business Workflow, SAP EarlyWatch, SAP ArchiveLink, BAPI, SAPPHIRE, Management Cockpit, SEM, are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other products mentioned are trademarks or registered trademarks of their respective companies.

## Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax
	Tip

## Contents

<b>Variant Configuration (LO-VC)</b>	<b>9</b>
<b>Defining a Material as Configurable</b>	<b>11</b>
<b>Material Master Data for Configurable Materials</b>	<b>12</b>
<b>Super BOM</b>	<b>13</b>
<b>Selecting BOM Items</b>	<b>14</b>
Classification as a Selection Condition	15
Selection from a Class Item	16
Selection from a Class Item: Example	17
Finding Objects in a Class Item	18
<b>Routings for Configurable Materials</b>	<b>19</b>
<b>Maintaining Characteristics for Configuration</b>	<b>21</b>
<b>Reporting: Characteristics and Values</b>	<b>23</b>
<b>Variant Classes</b>	<b>24</b>
<b>Defining Classes as BOM Items:</b>	<b>25</b>
<b>The Configuration Profile</b>	<b>27</b>
<b>Creating a Configuration Profile</b>	<b>29</b>
Class Assignment	32
Filters for BOM Explosion	34
Availability of Components	35
<b>Process Overview</b>	<b>36</b>
Process: Sales Order	37
Process: Order BOM	42
<b>Controlling the BOM Explosion</b>	<b>44</b>
No BOM Explosion	46
Single-Level BOM Explosion	49
Multi-Level BOM Explosion	51
<b>Interface Settings</b>	<b>53</b>
Defining Settings for the Language	54
Defining the Scope and Display Options for Characteristics	55
Defining Settings for Pricing	57
Defining Settings for Default Values	58
Defining Settings for the Configurator	59
Settings for Variant Matching	60
<b>Changing/Displaying/Deleting Configuration Profiles</b>	<b>62</b>
<b>Possible Combinations of Configuration Profiles</b>	<b>63</b>
<b>Dependencies</b>	<b>65</b>
<b>Global Object Dependencies</b>	<b>66</b>
Creating Global Dependencies	67
Changing Global Dependencies	68
Displaying Global Dependencies	69
<b>Local Object Dependencies</b>	<b>70</b>
<b>Maintenance Authorizations for Dependencies</b>	<b>71</b>
<b>Preconditions</b>	<b>73</b>
Example: Precondition for a Characteristic Value	74
Precondition for a Characteristic	76

<b>Selection Conditions</b>	<b>77</b>
Selection Conditions for a BOM Item and Operation	78
Selection Condition for a Characteristic	80
<b>Procedures</b>	<b>81</b>
Processing Sequence of Procedures	83
Inferring a Characteristic Value with Procedures (Example)	84
Built-In Function \$COUNT_PARTS	85
Built-In Function \$SUM_PARTS	87
Setting Default Values with Procedures	89
Deleting Default Values with Procedures	91
<b>Assigning Object Dependencies</b>	<b>92</b>
<b>Changing Master Data with Dependencies</b>	<b>96</b>
Reference Characteristics in Dependencies	98
Master Data References in Bills of Material	100
Master Data References in Task Lists	101
Changing the Weight in the Sales Order	103
Example: Shape and Size Variants	104
<b>Constraints</b>	<b>106</b>
Constraints: Referring to Objects	108
Constraints: Entering Conditions	111
Constraints: Restrictions	114
Constraints: Setting Values	116
Constraints: Restricting the Allowed Values	118
Creating a Dependency Net	120
Changing a Dependency Net	122
Displaying a Dependency Net	123
Networks in a Company	124
Relationship Between Operating System and Server	128
Relationship Between Operating System of Server and Operating System of Workstation	129
Relationship Between Company Network Server and Department Network Server	131
Relationship Between LAN Type and Server Processor	133
Dependency: Cable Type and LAN Type	134
<b>Actions (Obsolete)</b>	<b>136</b>
Inferring a Characteristic Value with Actions (Example)	137
<b>Where-Used List for Dependencies</b>	<b>138</b>
<b>Dependency List</b>	<b>139</b>
<b>Declarative Dependencies</b>	<b>140</b>
<b>Value Assignment Attribute for Characteristics</b>	<b>142</b>
Single-Value Characteristics	143
Multiple-Value Characteristics	144
<b>Restrictable Characteristics</b>	<b>146</b>
Restricting a Characteristic	148
Restricting Characteristics with a Variant Table	149
Restricting a Characteristic with a Table: Example	150
Restricting a Characteristic with IN	152
Assigning Values to Restrictable Characteristics	153
<b>Using Dependencies to Change how Characteristics are Displayed</b>	<b>154</b>
<b>Dependency Syntax: General Rules</b>	<b>156</b>
<b>Entering Characteristics and Characteristic Values</b>	<b>158</b>

Using Arithmetic Operations.....	160
Entering Intervals .....	162
Entering Comparisons.....	163
Object Variables .....	164
Built-In Condition SPECIFIED .....	166
Built-In Condition IN.....	167
Built-In Condition TYPE_OF .....	168
Variant Tables.....	169
Creating a Table Structure .....	171
Changing a Table Structure .....	172
Displaying a Table Structure.....	173
Value Assignment Alternatives.....	174
Multiple-Value Characteristics in Table Calls.....	176
Link to a Database Table .....	178
Linking a Variant Table to a Database Table .....	181
Transferring Variant Table Contents to Database Table .....	182
Maintaining Table Entries.....	184
Changing Table Entries .....	186
Displaying Table Entries.....	187
Decision Table .....	188
Tables in Actions and Procedures .....	189
Tables in Constraints.....	191
Tables in Preconditions.....	193
Tables in Selection Conditions .....	195
Creating Table Lists .....	197
User-Defined Functions.....	198
Creating a Function.....	200
Changing a Function.....	201
Displaying a Function .....	202
Interface of the Function Module.....	203
Function Call.....	205
Functions for Accessing the Dynamic Database.....	209
Creating a Function List .....	210
Variant Conditions .....	211
Variant Conditions in Purchasing.....	212
Maintaining Variant Conditions in the Info Record.....	214
Variant Conditions in Sales.....	215
Maintaining Variant Conditions in Sales .....	217
Assigning Variant Conditions Directly.....	218
Variant Conditions in Procedures .....	219
Variant Conditions with a Table.....	220
Pricing Factors .....	222
Dependency Group for Pricing .....	223
Material Variants.....	224
Maintaining Material Master Records for Variants.....	226

Bill of Material (BOM) .....	228
Task List .....	229
Creating Cross-Plant Material Variants .....	230
The Configuration Simulation .....	231
Simulating BOM/Task List Explosion .....	233
Selecting Configured Objects to Copy .....	234
Level of Detail for the BOM .....	235
Simulation of Costing .....	236
Creating a Routing with the Configuration Simulation .....	237
Creating a Routing from the Configuration Simulation .....	238
The Characteristic Value Assignment Screen .....	239
Configuring Objects .....	240
Processing Sequence for Dependencies .....	242
Explanation Functions for Value Assignment .....	243
Trace Function .....	244
Configuration Buffer .....	245
Interface Design – Overview .....	246
Defining an Interface Design .....	248
Maintaining an Interface Design .....	250
Defining the Sequence of Characteristics .....	252
Relevance to Printing of Characteristics .....	254
Specifying Enhancements in the Configuration Editor .....	255
The Configuration Result .....	257
Explanation Functions: Result Screen .....	259
Defining Scope and Display Options for the Result .....	260
Configurable Materials in Sales Documents .....	262
Item Categories for Configurable Materials .....	264
Graphic 1 .....	266
Graphic 2 .....	267
Graphic 3 .....	268
Graphic 4 .....	269
Graphic 5 .....	270
Graphic 6 .....	271
Graphic 7 .....	272
Variant Matching in the Sales Order .....	273
Transfer of Requirements for Locked Configurations .....	275
Changing Fields in a Sales Order .....	278
Low-Level Configuration .....	280
Configurable Materials in Purchasing .....	281
Displaying a Configuration Overview .....	283
Enhancements in Variant Configuration .....	284
ALE Transfer of Configuration Data .....	287
EDI for KMATs (Information on Creating Your Own) .....	291
Basic Type ORDERS02 .....	293
Segment Type E1CUREF .....	296
Segment Type E1CUCFG .....	297
Segment Type E1CUINS .....	298

<b>Segment Type E1CUPRT .....</b>	<b>300</b>
<b>Segment Type E1CUVAL .....</b>	<b>302</b>
<b>Creating an Order for Configurable Materials with EDI.....</b>	<b>303</b>
IDoc Structure .....	305
Example: Segment E1CUREF .....	307
Example: Segment E1CUCFG.....	308
Example: Segment E1CUINS .....	309
Example: Segment E1CUPRT .....	312
Example: Segment E1CUVAL .....	314
<b>Error Handling .....</b>	<b>316</b>
<b>Restrictions.....</b>	<b>317</b>
<b>Creating a Knowledge Base Object for the SCE .....</b>	<b>318</b>
<b>Creating a Knowledge Base Object.....</b>	<b>320</b>
<b>Creating a Runtime Version .....</b>	<b>322</b>
<b>Changing a Runtime Version .....</b>	<b>323</b>
<b>Loading Data for a Runtime Version .....</b>	<b>324</b>
<b>Creating a Database Schema for the SCE .....</b>	<b>331</b>
<b>OO Class .....</b>	<b>332</b>



## Variant Configuration (LO-VC)

### Purpose

Variant configuration is for manufacturing complex products. The manufacturer is always having to offer new variants of its products. Often, new variants are created by modifying existing product designs as you process the order. The important thing is to react quickly to customers' requirements.

The customer determines the features of the product. A customer buying a car, for example, can choose the features of the car and combine these features as required.

The product configurator improves information exchange between sales, engineering, and production. Variant configuration helps the customer or salesperson to put together specifications for the product and ensure that the product can be produced from these specifications. It also ensures that production costs do not overstep the mark.

### Integration

Variant configuration is integrated in the following applications:

- CA Classification
- LO Material Master
- PP Bill of Material
- PP Routings
- PP–PI Master Recipes
- SD Sales
- SD Conditions
- MM Purchasing
- CO Costing
- PP Material Requirements Planning (MRP)
- PP Production Orders

You can configure the following objects:

- Materials
- Standard networks in the Project System.
- PM General maintenance task lists
- Model service specifications

### Features

You do not need to create a separate material for each variant of a product in your company. You can use one **configurable material** to cover all variants. You create a **super BOM** and a **super routing** for this material, containing all the components and operations for producing all variants of the product.

## Variant Configuration (LO-VC)

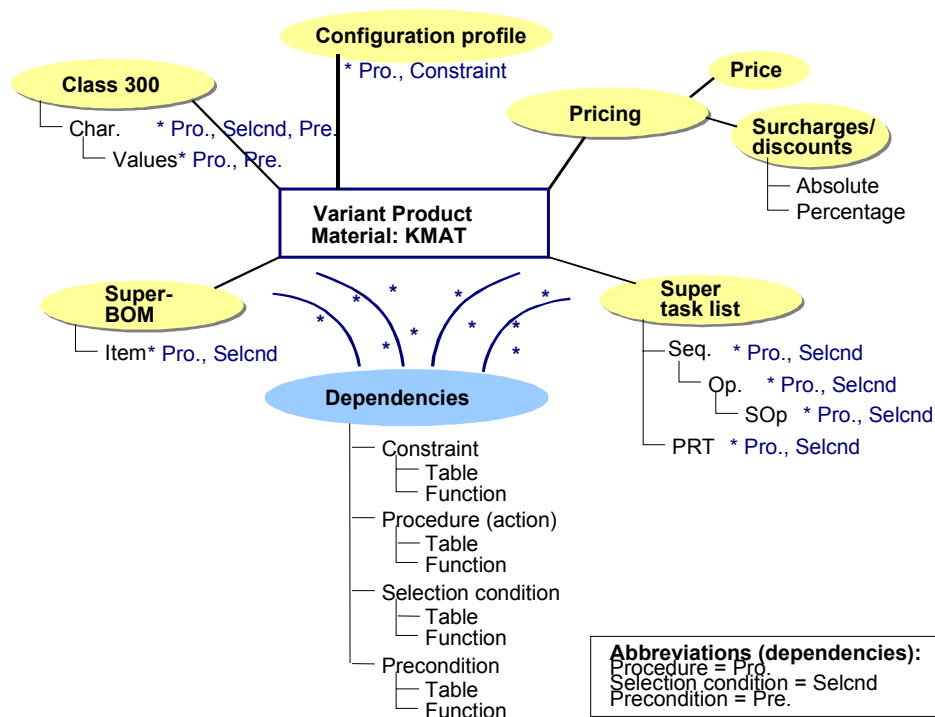
To define the features of a configurable material, you use **characteristics**. To enable you to use characteristics to configure a material, you assign the material to a class of **class type 300**. Possible characteristics of a car, for example, are model, country, color, and engine. The customer can choose from among different options for each characteristic (**values**).

The only limitations are combinations of features that are not possible for either technical or marketing reasons. You cannot combine all types of car engine with all types of transmission, for example, or certain types of upholstery are only available for more expensive models. In Variant Configuration, you use **dependencies** to control this. This prevents combinations of options that are not allowed. Dependencies also select exactly the right BOM components and operations to produce a variant.

Each configurable object must have a **configuration profile**. The configuration profile for a material controls the configuration process in the sales order.

You can make the price of a variant dependent on the characteristic values assigned (**Pricing**). You can use **variant conditions** to define surcharges and discounts for a variant.

For variants that are required frequently, you can create **material variants**, which can be produced without a sales order and kept in stock. When you receive a sales order, you can check whether the variant required is in stock, so that you can deliver immediately.



## Defining a Material as Configurable

### Use

If a material is configurable, you can:

- Select one or more alternatives, even though they are represented by only one configurable material
- Configure the BOM and routing of the material Maintain dependencies for BOM items and operations, so that they are only selected for certain variants
- Maintain a [Configuration Profile \[Page 27\]](#) with a class assignment for the material, where you define central settings for the configurable object

### Prerequisites

Before you can configure a material, the material master record must have the *Material is configurable* indicator selected in its *Basic data*.

- You can create a material using a material type that has the *configurable* indicator defined in Customizing. This means that all materials created with this material type are configurable. In the standard system, material type KMAT is defined for this purpose.
- You can define individual materials of other material types as configurable. To do this, set the *Material is configurable* indicator in the *Basic data* of the material master record.



If you want to configure a material, you must create it with a material type that has the *Classification* view defined as allowed in Customizing. Otherwise, the assignment of a material to a class cannot be processed correctly.

For information on how to create a material master record for a configurable material, see the SAP Library *LO Managing Material Master Data*.

#### See also:

[Material Master Data for Configurable Materials \[Page 12\]](#)

## Material Master Data for Configurable Materials

## Material Master Data for Configurable Materials

You use central maintenance functions to create material master records for configurable materials. There is some data that you need to maintain specifically for configurable materials:

### Basic Data

Material is configurable indicator	X
------------------------------------	---

### Sales

Item category group	0002 or 0004 See <a href="#">Item Categories for Configurable Materials [Page 264]</a>
Delivering plant	

### MRP

Strategy group	For example, 25
	See SAP Library <i>PP Demand Management</i> <a href="#">Strategies for Configurable Materials [Ext.]</a>
MRP type	For example, PD Not ND
	See SAP Library <i>PP Demand Management</i> <a href="#">MRP Procedures [Ext.]</a>
MRP lot size	EX
	See SAP Library <i>PP Material Requirements Planning</i> <a href="#">Calculating Procurement Quantity [Ext.]</a>
Availability check (checking group)	02
	See SAP Library <i>PP Demand Management</i> <a href="#">Scope of the Availability Check [Ext.]</a>
Individ./collective	1

### See also:

SAP Library *LO Managing Material Master Data*

## Super BOM

### Use

The bill of material (BOM) of a configurable material contains all the components that are required to manufacture the material. The BOM contains components that are only used in specific variants (variant parts), as well as components that are used in all variants (non-variable parts).

This is why BOMs for configurable materials are known as super BOMs.

### Features

When you maintain BOMs for *configurable* materials, there are additional functions:

- You can assign object dependencies to the BOM items for a configurable material.  
You can assign the following dependency types to BOM items:
  - Selection conditions  
To ensure that variant parts are selected when they are needed in a variant, you assign selection conditions to the variant parts.  
See [Selection Conditions \[Page 77\]](#) and [Selection Conditions for a BOM Item and Operation \[Page 78\]](#)
  - Procedures (actions)  
You can use procedures and actions to change field values in a BOM item, such as the component quantity.  
See [Reference Characteristics in Dependencies \[Page 98\]](#) and [Master Data References in Bills of Material \[Page 100\]](#)
- You can use classes as BOM items for configurable materials, as well as the other types of component. When you configure the material, the characteristic values you assign can trigger the replacement of the class by a material or a document that is classified in the class. You can use a class item instead of using several BOM items with selection conditions.
- You can use the classification data of a material or document as a selection condition.
- The BOM can contain components that are configurable materials. These configurable materials have item category N (non-stock material).

#### See also:

PP Bills of Material

[Maintaining the Create Material BOM Initial Screen \[Ext.\]](#)

[Creating New Items \[Ext.\]](#)

## Selecting BOM Items

## Selecting BOM Items

### Use

There are three ways of selecting variant parts in the BOM of a configurable material:

- You enter the variant part as a BOM item and assign selection conditions to it
- You classify the variant part in a class and enter the class as a class item in the BOM.
- You use the classification data of a material or document as a selection condition.



A bike has different rear lights:

- Dtoplight Plus
- Toplight
- FER

You have the following options for controlling which rear light is selected:

- a You can create BOM items for the individual rear lights, and use selection conditions to define when each item is selected.
- b You can create a class of class type 200 or 300 and classify the three rear lights in the class. You enter the class as a BOM item in the BOM.  
When you configure the material, the class is replaced by one rear light.
- c The rear lights were classified in the classification system (in a class of class type 001, for example). Characteristic REAR\_LIGHT\_TYPE has been defined for the rear lights. You assign this characteristic to the variant class.

You enter three items in the BOM for the rear lights. For each item, you define that its classification is used as a selection condition. If you set a value for characteristic REAR\_LIGHT\_TYPE when you configure the bike, the system looks for a rear light that has this value, and selects a rear light on this basis.

## Classification as a Selection Condition

### Use

You can classify the materials or documents that you enter as components in a BOM in the Classification System. The bolts that are selected for different variants can be classified in class BOLTS and entered as BOM items. You can use the classification data in variant configuration as a selection condition.

In BOM maintenance, you just define in the BOM item that the item's classification controls whether it is selected – there is no need to enter a selection condition. You can use the classification in any class type.

### Prerequisites

The characteristics of the class in which the material or document is classified must all be assigned to the configurable material. The value assigned in configuration must be identical to the classification value before the material or document is selected.



Class BOLTS in class type 001 has characteristics LENGTH and HEAD\_SHAPE. To use the classification of bolts as a selection condition, you must assign both characteristics to the variant class. Before a bolt can be selected, both characteristics must have both classification values assigned in configuration.

### Activities

On the detail screen of the BOM item in BOM maintenance, enter the class type of the class in which the material is classified, and select *Selection condition*.

#### See also:

[Selecting BOM Items \[Page 14\]](#)

## Selection from a Class Item

## Selection from a Class Item

### Use

You have various options for selecting a specific material or document from a class item (sometimes called “specializing” a class item).

- The class item is automatically replaced according to the characteristic values assigned when the material is configured.
- This means that object dependencies for the class item must make values for the characteristics of the class known.

If the system finds no values for the characteristics in the class item, first it looks for the values in the superior assembly, then if it finds none there, it checks the header material.

- In a multilevel, interactive configuration, the class item can be replaced manually by searching for an object.

However, this is only possible in the simulation, or in configuration processes *Order BOM* and *Sales order* (with the setting *Manual changes allowed*).



Components that are automatically selected from a class item cannot be changed manually in order BOM processing or set processing.

### Required Component

If you define in the class or the BOM item data that a component must be selected (*Required component*), the class item is marked as inconsistent until it is replaced by a material or document.

#### See also:

[Defining Classes as BOM Items \[Page 25\]](#)



## Selection from a Class Item: Example

1. Create characteristic REARLIGHT\_TYPE, with values Dtoplight Plus, Toplight, and FER.
2. Create class REARLIGHT with class type 200.

Define the following additional data for the class:

- The class can be used in bills of material (BOMs).
- The base unit of measure is piece.
- The item category for the resulting material is L (stock item).

3. You have created 3 materials – R1220, R1221, and R1222.

You classify these 3 materials in class REARLIGHT as shown in the following table:

Material	Value
R1220	Dtoplight Plus
R1221	Toplight
R1222	FER

4. You enter the class as a BOM item in the BOM for configurable material BIKE.



When you enter the BOM item, enter item category K first. You see a dialog box in which you must enter the class name and class type.

The class name for the item is not shown on the material item overview.

5. Assign characteristic REARLIGHT\_TYPE to the variant class of configurable material BIKE.

The characteristic is now assigned to both the class item and the configurable material.

6. When you configure the BIKE, you can assign a value to characteristic REARLIGHT\_TYPE.

Depending on the value you assign to the characteristic, the relevant material is selected from the class item.

For example, if you assign value 'Toplight' to characteristic REARLIGHT\_TYPE, the class item is automatically replaced by material R1221.

---

Finding Objects in a Class Item

## Finding Objects in a Class Item

### Procedure

You can start the function for finding objects in a class item from the result screen, or in a multi-level, interactive configuration, you can select the class item from the *Config structure*:

1. Select the class item and choose *Find objects*.

You see the characteristics of the class. You can enter characteristic values as search criteria.

2. Choose *Find object* again.

You see a list of objects that match your search criteria.

3. Select the object you want to copy and choose *Copy objects*.

However, this is only possible in the simulation, or in configuration processes *Order BOM* and *Sales order* (with the setting *Manual changes allowed*).

### Result

On the result screen, the class item is replaced by the object you selected.

## Routings for Configurable Materials

### Use

A routing (or task list) for a configurable material contains all the operations, operation sequences, and production resources/tools (PRTs) that are required to manufacture all variants of a configurable material.

Routings (task lists) for configurable materials are known as super task lists.

When you configure the material, you assign characteristic values that are used in production to determine the operations required.

In the processing industries, master recipes are used instead of task lists. Master recipes contain all the recipe objects that are required for all product variants.

### Features

To maintain task lists for configurable materials, choose *Routings* → *Routing* → *Create*. You create the routing in the same way as for other materials.

You can assign dependencies to the operations, operation sequences, and PRTs in routings for configurable materials.

You can assign dependencies to the phases, operations, BOM items, and secondary resources in master recipes.

You can assign the following dependency types:

- Selection conditions

To ensure that operations, operation sequences, and PRTs are selected when they are needed in a variant, you assign selection conditions to them.

Selection conditions also ensure that the correct objects in master recipes are selected.

See [Selection Conditions \[Page 77\]](#) and

[Selection Conditions for a BOM Item and Operation \[Page 78\]](#).

- Procedures (actions)

To change fields in operations and PRTs, such as the standard values, you use procedures or actions.

In master recipes, you can change fields in operations, phases, secondary resources, and BOM components.

See [Reference Characteristics in Dependencies \[Page 98\]](#) and [Master Data References in Task Lists \[Page 101\]](#)

#### See also:

PP Routings

[Creating a Routing \[Ext.\]](#)

[Creating Local Object Dependencies for an Operation \[Ext.\]](#)

[Creating Global Object Dependencies in an Operation \[Ext.\]](#)

---

## Routings for Configurable Materials

PP–PI Master Recipes

[Creating Master Recipes \[Ext.\]](#)

[Editing Configurable Master Recipes \[Ext.\]](#)

[Editing Global Object Dependencies \[Ext.\]](#)

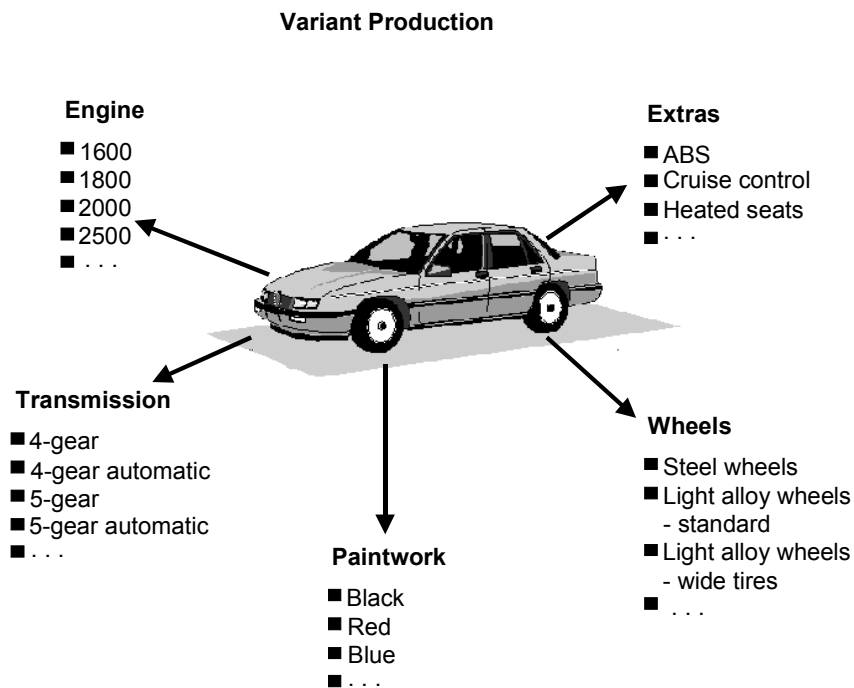
[Editing Local Object Dependencies \[Ext.\]](#)

## Maintaining Characteristics for Configuration

### Use

When you create a sales order for a configurable material, this sales order must describe precisely how the product being ordered is to look. This description comes from characteristics and characteristic values.

A car, for example, has a large number of options that need to be described.



The graphic shows the individual product options such as engine, gears, paintwork, mapped as characteristics. You define values for characteristics, to allow you to select specific options, such as 'black' for paintwork.

To create characteristics, you use the standard functions in the classification system menu (see the SAP Library *CA Cross-Application Components* → *Classification (CA-CL)* → *Characteristics (CA-CL-CHR)* → [Creating, Changing, and Displaying Characteristics \[Ext.\]](#)).

### Features

You can assign object dependencies to characteristics and characteristic values, to ensure that the values assigned are complete and consistent (see *Characteristics (CA-CL-CHR)*: [Creating Dependencies for a Characteristic \[Ext.\]](#) and [Maintaining Dependencies for a Characteristic Value \[Ext.\]](#)).

You can define characteristics as restrictable, specifically for variant configuration. The values of a restrictable characteristic can be restricted during configuration to certain allowed values (see [Restrictable Characteristics \[Page 146\]](#)).

---

**Maintaining Characteristics for Configuration**

## Reporting: Characteristics and Values

You can display all the characteristics that describe a configurable material. You also see information on whether an entry is required for a characteristic, whether a default value is defined for a characteristic, and whether dependencies are allocated to a characteristic.

1. Choose *Environment* → *Reporting* → *Chars and Values* from the variant configuration menu.

You see the initial screen. Enter the material for which you want to execute this function. You can enter the following data:

- The date on which you want to execute the reporting function.  
If the material has been processed with engineering change management on different dates, different characteristics may be assigned to the material on different dates.
- The language in which you want the characteristics displayed.  
In characteristics maintenance functions, you can enter language-dependent descriptions for characteristics and for values with CHAR format.
- You can define that characteristics are displayed with values.
- You can define that the status of dependencies is checked. If a dependency that is not released is allocated to a characteristic or characteristic value, the indicator for dependencies is selected in the result of the report.

Once you have defined your settings, choose *Execute*.

2. You see the configurable material and the class to which the material is allocated. Below it, you see a list of the characteristics of the material. You can print this list.

If object dependencies are allocated to a characteristic or characteristic value, the *O* indicator is selected. If you are interested in the status, released and locked dependencies are displayed differently.

If a characteristic is required entry, the *R* indicator is selected.

If a value is defined as a default value, the *D* indicator is selected.

You can select a characteristic and display detailed information on its format and attributes.

---

Variant Classes

## Variant Classes

In variant configuration, a class is used to hold the characteristics that describe a configurable material. By linking the class to the configurable material, you allow the material to be configured using the characteristics of the class.

You can only use a class in variant configuration if the class has a class type that supports variant configuration. In the standard R/3 System, the class type for variants is class type **300**. However, in Customizing for *Classification*, you can define other class types for variant configuration. In the step *Define class types*, you can set the *Variant class type* indicator for a class type.

Once you have created a variant class, you can do the following, as for any other class:

- Classify materials in the class  
These materials do not have to be configurable materials.
- Set up a class hierarchy  
This allows you to use inherited characteristics and restrict characteristic values.
- Use the class to find objects

Class type **300** has 2 other special attributes, in addition to the *Variant class* indicator:

- In classes of this class type, you can classify objects of different object types. This means that all the objects linked to the configurable material are classified in one class.  
For example, if you create a sales order for a configurable material, this order is automatically classified in the variant class.
- You can also use classes of class type **300** as BOM items. For more information, refer to [Classes as BOM Items \[Page 14\]](#).

You define these two settings in Customizing for *Classification*, step *Define class types*.



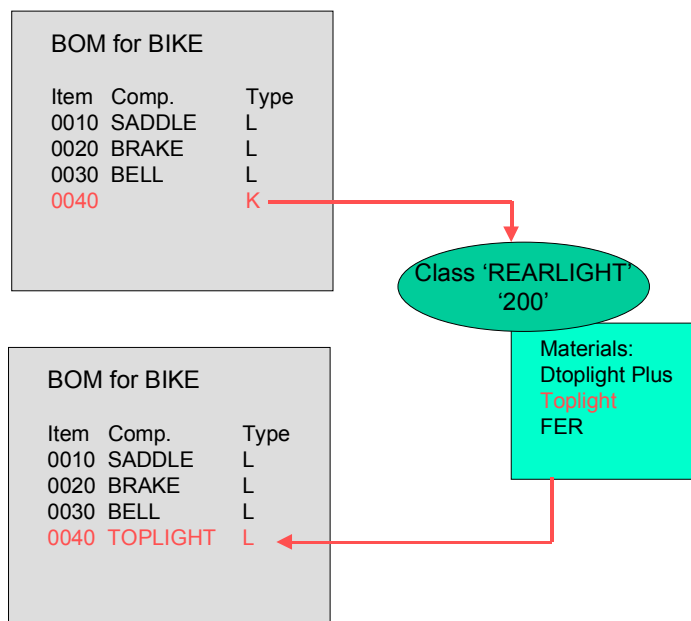
## Defining Classes as BOM Items:

### Purpose

You can use class items to control the selection of variant parts in a BOM. Variant parts are classified in a class, and the class is entered in the BOM as a class item. When you configure the material, the class is replaced by a suitable variant part.

Class items can help you to simplify maintenance of BOMs for configurable materials:

- You only enter one item for the class, instead of several items for the individual variant parts.
- You do not need to create and assign selection conditions.
- You can classify additional objects in the class at any time, without having to change the BOM.



When you configure the material, the class item is replaced by a specific component.

### Prerequisites

You can only use classes for classifying the following objects as BOM items:

- Materials
- Documents

The class type determines whether a class can be used as a class item in BOMs. In Customizing for the *Classification System*, the *Class node* indicator must be set for the class type. In the standard system, class types 200 and 300 are defined for materials, and class type 201 is defined for documents.

**Defining Classes as BOM Items:**

Configurable materials can be assigned to classes of class type 300. If a class item is replaced by a configurable material, this material is added to the configuration structure automatically.

**Process Flow**

1. Create characteristics that describe the objects.
2. Create your class using a class type for class items.  
Maintain the additional data for the class (see the SAP Library *CA - Cross-Application Components* → *Classification (CA-CL)* → *Classification (CA-CL)* → [Additional Data \[Ext.\]](#)).
3. Assign objects to the class and assign values to the characteristics of the class for each object.
4. Enter the class as a class item in the BOM of the configurable material – see the SAP Library *Bills of Material (PP-BD-BOM)*.
  - [Detail Screens For A Class Item \[Ext.\]](#)
  - [Entering Class Items \[Ext.\]](#)

**See also:**

[Selection from a Class Item \[Page 16\]](#)

# The Configuration Profile

## Purpose

You maintain configuration profiles for configurable objects to define central settings for configuring the object.

You can create several configuration profiles with different settings for an object. If an object has several configuration profiles, you must select a profile during configuration. You can only make changes to a configuration with the profile that you first used to configure the object.

## Integration

You maintain configuration profiles for all configurable objects:

- Configurable materials
- Configurable standard networks
- Configurable general maintenance task lists
- Configurable model service specifications

## Features

You define different settings in configuration profiles for materials than for other configurable objects.

- You use the configuration profile to assign the configurable object to one or more variant classes. This links the object to the characteristics of the class for configuration.



Note that the class is then assigned to the configurable object, not the profile. The profile only allows you to go to classification.

- You can define settings that affect the display options and scope of characteristics on the value assignment screen. You define these settings for each object, and they apply to the object wherever it is used. However, you can overwrite these settings for your user in the configuration editor.
- You can use an interface design to group characteristics together and define a sequence on the value assignment screen. You need to assign a name to an interface design to enable further maintenance in the configuration simulation.
- You can use the configuration profile to assign dependencies to a configurable object. You can only assign dependency nets to a configuration profile. If you assign actions and procedures to the configuration profile, you can manage them more easily because they are all in one place.
- For materials, you must also maintain configuration parameters for BOM explosion. Depending on the configuration parameters you set, other fields in the configuration profile are hidden or shown.
- You determine the status of the configurable object.

---

**The Configuration Profile**

You can also define settings for the status in Customizing. In Customizing, choose *Configuration Profile → Configurable Objects → Material → Maintain Status*.

## Creating a Configuration Profile

### Procedure

From the Variant Configuration menu, choose *Configuration profile* → *Create*.

You see a dialog box. Select the object for which you want to create a configuration profile.


Enter the name of the object.

Confirm your entries.

3. You see the profile overview.
4. Enter a profile name and the class type whose classes you want to use for configuration. In Customizing for the *Classification System*, the class type must be defined as a variant class type.
5. Choose *Goto* → *Class assignments* or the *Class assignments* pushbutton to assign the object to a class. (You can also specify the class assignment in *Materials Management* by choosing *Create/Change Material*, or in the Classification System by choosing *Assign Object to Class*.)
  - You see the classification screen, where you enter the class.
  - Define allowed values for the configurable object if required.



You can only release a profile if you have assigned the configurable object to a class. (see also [Class Assignment \[Page 32\]](#))

6. To go to the detail screen, where you define further settings, choose the  *Profile detail* pushbutton or double-click on the profile.

### Optional Settings on the Profile Overview

- **Organizational Areas**

You can specify organizational areas for configuration. If you restrict a profile to specific organizational areas, you only see the characteristics that are relevant to your area when you configure the object.

You can change how characteristics are displayed according to organizational area for your user on the value assignment screen (see [Organizational Areas \[Ext.\]](#)).

- **Priority of Configuration Profiles**

You can create several configuration profiles for a configurable material. The profile with the lowest number has the highest priority.

If you define several profiles for an object, you see a dialog box for selecting a profile when you start configuration. The profile with the highest priority is at the top of the list.

The priority is also relevant if you use Application Link Enabling (ALE) or intermediate documents (IDocs) to run configuration, rather than dialog mode. In this case, the profile with the highest priority is selected for the object.

## Creating a Configuration Profile


Profiles that have no priority are at the top of the list, because they automatically have priority 0.

## Profile Detail

### Configuration Profiles for Materials

The configuration profile has several screens. Depending on what you enter for the BOM explosion and configuration process, fields are shown or hidden.

The parameters you maintain for a material apply to the material as a header material in a BOM structure. You cannot define separate settings for use of a configurable material as an assembly in a BOM.

- By choosing the  *Assignments* pushbutton, you can assign dependencies to the configuration profile. You also see this pushbutton on the basic data tab, once at least one dependency is assigned.

### Basic Data Tab

- In the basic data, you see the profile overview data.
- You can determine whether the configuration process starts with a *Start logo*. To do this, you assign the document you want to display, such as a graphic showing the product you want to configure, to the variant class.

### Configuration Initial Screen Tab

#### *Configuration parameters*

You specify the parameters for BOM explosion and the configuration process in the sales order.

- You must enter a BOM application, unless you select *BOM explosion: None*.
- You can enter a level of detail for a multi-level configuration. You can display configurable assemblies only in the BOM explosion. This improves system performance.
- You can define a filter.
- You can set the indicator for an availability check on assemblies.

#### *Userinterf*

Under *Userinterf*, you maintain the settings for the configuration editor user interface.

- You can specify an interface design to group characteristics together on the value assignment screen.
- You can choose *Settings* to define object-specific settings for functions in the configuration editor. These settings are defaults for configuration, and can be overwritten for your user in the configuration editor.
- For all configuration parameters except *BOM explosion: None*, you can select screens for the configuration editor and define which screen configuration starts with. The start screen must be one of the allowed screens.

You can select the indicator for the configuration browser independently of the other start screens, because the browser is an additional screen section.

### Configuration Profiles for Objects Other than Materials

You see the basic data, where you see the profile overview data.

## Creating a Configuration Profile

You can define object-specific settings for displaying characteristics and characteristic values in the configuration editor. These settings are defaults for configuration, and can be overwritten for your user in the configuration editor.

### See also:

On configuration parameters:

[Controlling the BOM Explosion \[Page 44\]](#)

[No BOM Explosion \[Page 46\]](#)

[Single-Level BOM Explosion \[Page 49\]](#)

[Multi-Level BOM Explosion \[Page 51\]](#)

[Process: Sales Order \[Page 37\]](#)

[Process: Order BOM \[Page 42\]](#)

[Filters for BOM Explosion \[Page 34\]](#)

[Availability of Components \[Page 35\]](#)

On settings:

[Defining Settings for the Language \[Page 54\]](#)

[Defining the Scope and Display Options for Characteristics \[Page 55\]](#)

[Defining Settings for the Default Values \[Page 58\]](#)

[Defining Settings for Pricing \[Page 57\]](#)

[Defining Settings for the Configurator \[Page 59\]](#)

[Settings for Variant Matching \[Page 60\]](#)

## Class Assignment

# Class Assignment

## Use

Before you can configure an object, you need to assign the object to a class whose class type supports variant configuration. For materials and other configurable objects, this is class type 300 in the standard system, except for model service specifications, which have class type 301. When you assign the object to the class, you can use the characteristics of the class to describe the object.

## Prerequisites

In Customizing for the *Classification System*, the *Variant class type* indicator must be set for the class type.

## Features

### Class Assignment Pushbutton

The assignment to a class is not the same as any classification. The class is just a container for the characteristics that are required for configuring the object.

If you set values, this has the effect of restricting the allowed values for configuration, rather than assigning values as in classification. You can assign several values to single-value characteristics.

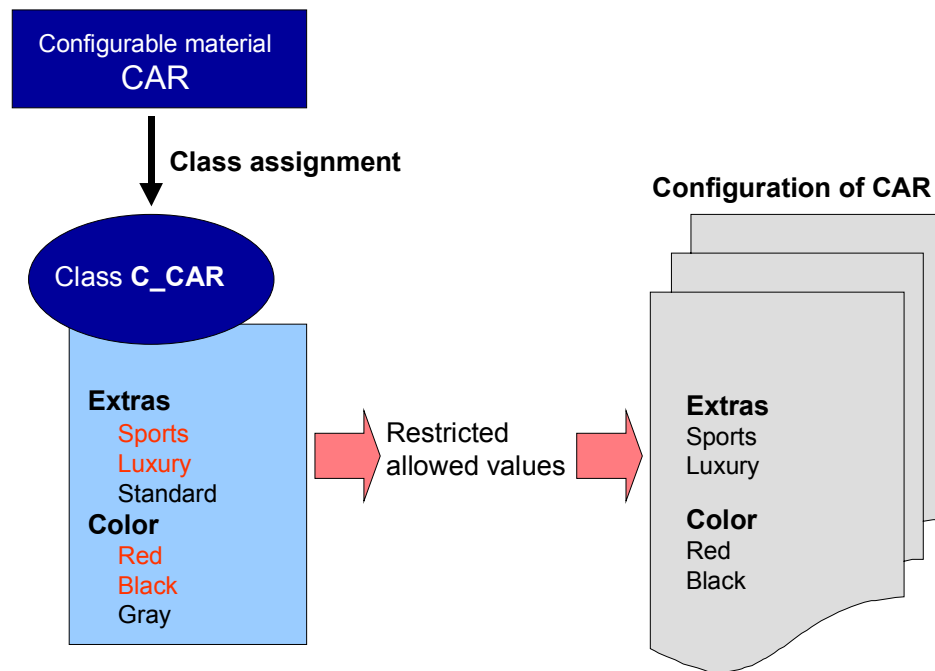
The values you set are not default values. During configuration, you only see the values you set in the class assignment.



Class C\_CAR has characteristics EXTRAS and COLOR. You assign configurable material CAR to class C\_CAR. You restrict characteristic EXTRAS to values 'Sports' and 'Luxury', and characteristic COLOR to values 'Red' and 'Black'.

When you configure the car, you only see the two values you set for both COLOR and EXTRAS, because you have restricted the allowed values in the class assignment.





To allow all values in configuration, do not set values in the class assignment.

### Multiple classification

If the class type allows multiple classification, you can classify the configurable object in several variant classes. When you configure the object, you see the characteristics of all classes of a class type to which the object is assigned.



If you only assign a configurable object to an additional class later on, and configurations (such as sales orders) already exist, you can no longer delete the assignment to the additional class once you have saved it.

The sequence of classes has no influence on the sequence of characteristics on the value assignment screen.

#### See also:

*SAP Library → CA Cross-Application Components → CA Classification*

## Filters for BOM Explosion

### Use

By defining a filter in the configuration profile, you can determine the scope of the BOM items to improve system performance when you explode the BOM. The filter is active in high-level configuration, in result-oriented BOMs, and in SET processing.

### Restrictions

You cannot define filters if you select the *BOM explosion* setting *None*.

### Features

You can define the following filters:

- Object type
  - Class, material, document, text
  - In the standard system, all object types are selected and therefore exploded in the configuration. Deselect the object types that you do not want to be displayed.
- Item category, for example, stock or non-stock item
  - All item categories in the configuration are exploded in the standard system. Remove the selection for the item categories you do not want to be displayed.
- Item status
  - You maintain the status of a BOM item in maintain BOM dependent on its usage.
  - All items are displayed regardless of their item status in the standard system. However, only the items with this status are displayed when you select specific item statuses. Items are not displayed that do not have the selected status.
- Sort string
  - You can assign sort strings for BOM items in maintain BOM. You can restrict the display of the BOM items by using these sort strings.
  - Only items that carry sort strings are checked and only those that match are displayed. Items that have no sort string are always displayed.

#### See also:

[Controlling the BOM Explosion \[Page 44\]](#)

## Availability of Components

### Use

If you select *Component availability* in the configuration profile, by choosing the *Confign initial screen* tab, then the *Configuration parameter* tab, you see the *Availability* pushbutton on the value assignment screen in the configuration editor. You can use this pushbutton to check the availability of the components that are selected according to the values you assign. The entire BOM is checked and exploded.

For this reason, in a SET structure we advise you to check availability of the individual materials (*Process setting Sales order*), not for the header material.

The availability check is just a snapshot, telling you whether the materials required are in stock at this moment. Several users can access the same material at once. This means that supply problems can sometimes be overlooked.



Only 2 pieces of a material are in stock, but the material is used in 3 BOMs. The availability check does not detect a supply problem. The availability check for all 3 BOMs shows 2 pieces in stock.

### Features

On the value assignment screen, you can define additional settings for the availability check:

- You can specify that a list of all components is shown with details of their availability.
- You can specify that only missing parts are shown in the list.

## Process Overview

### Purpose

There are different processes for configurable materials in sales documents. You can specify these processes on the *Config initial screen* tab in the configuration profile, by choosing the *Config parameters* tab. The processes are described in the following scenarios:

- **Plnd/Prod. Order:**

For information on this scenario, see:

[No BOM Explosion \[Page 46\]](#)

[Single-Level BOM Explosion \[Page 49\]](#)

[Multi-Level BOM Explosion \[Page 51\]](#)

You can use this processing type to describe variant products whose configurable materials are assembled using planned and production orders. The bill of material (BOM) can have single-level, multi-level, or no explosion.

- **Sales Order (SET)**

You can use this processing type to describe variant products that comprise salable configurable materials. These products are supplied together, but are not assembled in a production order. Only sales-relevant BOM items are exploded in the sales order.

For information on this scenario, see [Process: Sales Order \[Page 37\]](#).

- **Order BOM**

You use this processing type if you want to make customer-specific changes to the BOM of a material that you configure in the sales order. In the sales order, you assign values to the characteristics of the header material, but the BOM is not exploded in the sales order.

For information on this scenario, see [Process: Order BOM \[Page 42\]](#).

## Process: Sales Order

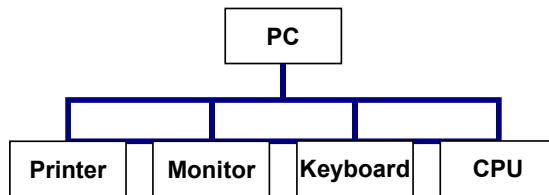
### Purpose

You can use this processing type to describe variant products that comprise salable configurable materials. These products are supplied together, but are not assembled in a production order.

This processing form means that in the sales order you can only process sales-relevant BOM items.

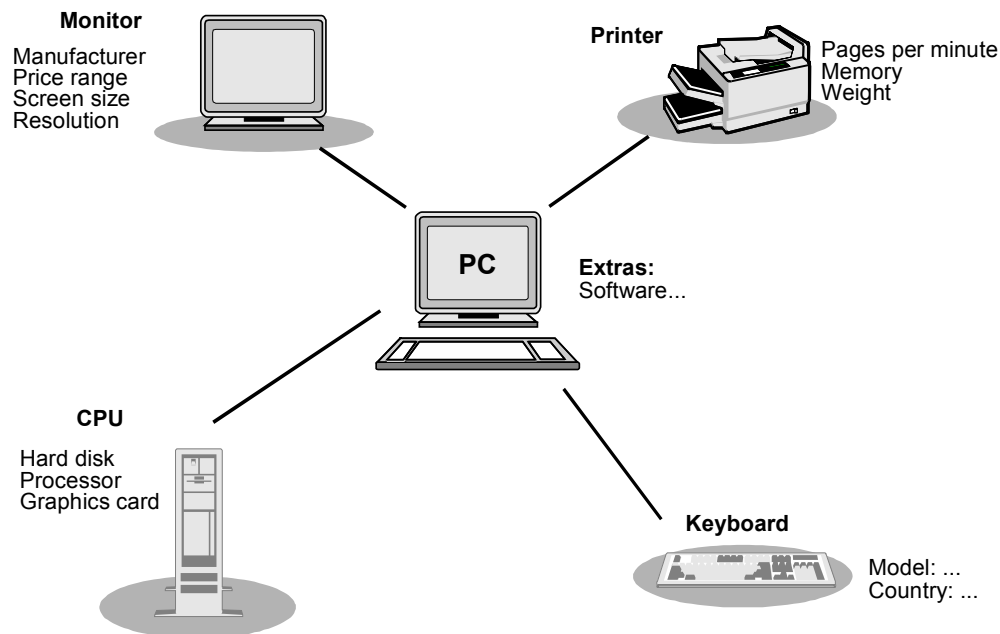
You can use the [item category group \[Ext.\]](#) to define that requirements transfer and pricing are at component level.

For example, a PC is made up of the components monitor, CPU, keyboard, and printer. These components are all salable materials that can be manufactured separately. In the sales order, you want to see the individual components that make up the computer as order items, as well as the whole computer. In sales order processing, these products are grouped together in a SET. For this reason, this type of processing is also referred to as **set processing**.



Sold-to party		XX
Requested delivery date		....
Item	Material	Pieces
10	PC	1
20	Printer	1
30	Monitor	1
40	Keyboard	1
50	CPU	1

The individual components of a BOM can be configurable products that are described by characteristics of their own. You must create a configuration profile for each of these materials, in order to assign the materials to a class. The superior configurable material only has a few characteristics that are relevant to all components.

**Process: Sales Order**

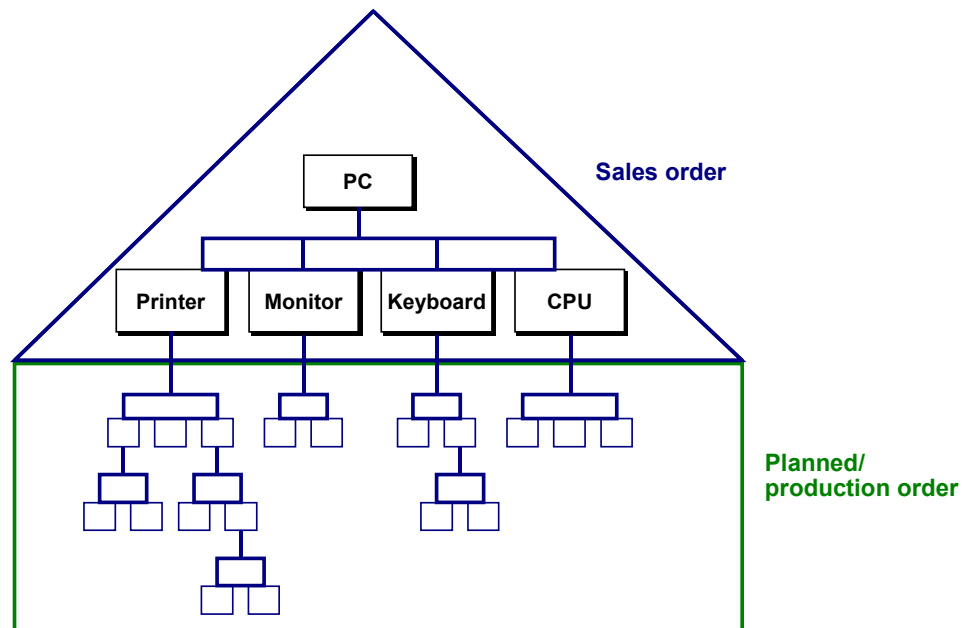
In addition to the process, the BOM explosion settings for the header material must be specified in the configuration profile.

- For a single-level SET structure, it is sufficient to define a single-level BOM explosion in the configuration profile of the PC.
  - *BOM explosion: single-level* setting and process *Sales order*.
- If the BOMs of the assemblies (such as the printer) contain other configurable materials, you need the multi-level BOM explosion setting for the PC:
  - *BOM explosion: multi-level* setting and process *Sales order*.

If the header material supports multi-level BOM explosion, the configuration parameters of the assembly determine whether the BOM is exploded in the sales order.



If you only want the assemblies of the header material to appear in the sales order, and the components of subordinate assemblies are relevant to production, you can select the setting *BOM explosion: Single-level* and process *Sales order* for the configuration profile of the header material. The sales-relevant BOM of the header material is exploded in the sales order, and the production-relevant BOM is exploded in the planned order or production order.



## Constraints

You can use constraints to infer values between configurable assemblies (for example, the printer and the CPU).

You can also use constraints to pass values from the PC to the assemblies, or from the assemblies to the PC.

## Object Variable \$PARENT

If you define selection conditions for the BOM items of a subordinate configurable assembly, and these selection conditions refer to the characteristics of the assembly, you must use the object variable \$PARENT to refer to the characteristics.



The selection conditions for BOM components of the printer must refer to the characteristics of the printer with object variable \$PARENT – for example, `$PARENT.Pages/min = '3'`.

## Prerequisites

- **Settings in the Configuration Profile**

Before the components of a configurable material can appear as a set in the sales order, the parameters *BOM explosion: Single-level* or *Multi-level* and the process *Sales order* must be selected in the configuration profile of the material.

- **Settings in BOM Maintenance**

Only BOM items that are defined as relevant to sales and distribution appear in the sales order.

**Process: Sales Order**

The BOM must be created with a usage that supports sales-relevant BOM items, and you must define the BOM items as sales-relevant by choosing *Item* → *Status/long text* (see [Controlling the BOM Explosion \[Page 44\]](#)).

- **Settings in the Material Master**

Sales data must be maintained in the material master of the individual materials in the BOM.

If you want requirements transfer and pricing at component level, the configurable material must have item category group 0004.

- **Allowing Manual Changes**

You can also change the BOM items of the header material to suit customer requirements. The change is copied to the sales order item. However, you can only change the BOM items if the *Manual changes allowed* indicator is set in the configuration profile of the header material. You can only change BOM items that are relevant to sales.

If manual changes are allowed, you can delete items from the BOM or insert items in the BOM. You can also make changes to existing items (for example, you can change the quantity).

## Process Flow

1. Create a sales order and enter the configurable material as an order item.
2. You see the configuration editor, where you assign values to the characteristics of the header material. You can then display the result of the BOM explosion by choosing *Result*.

You see all BOM components that are selected according to the values assigned, **and** are relevant to sales.

3. You can also configure the configurable assemblies. You can use constraints to infer characteristic values between assemblies. Once you have configured all materials, you return to the order item entry screen.
4. All BOM items that were selected and are relevant to sales are displayed as sub-items for the main item in the sales order.

For more detailed information on settings for item categories for transferring requirements and pricing, see [Item Categories for Configurable Materials \[Page 264\]](#).

## Note

If you select the process *Sales order* in the configuration profile, this may lead to problems in costing. You need to take note that each configured instance that has the *Sales order* setting represents a configuration of its own, so that each configurable material with this setting has its own CBase. The use of object variable \$ROOT in dependencies can easily lead to errors here.



Example

Item in the Sales Order	Material	Material Type	Settings in the Configuration Profile
10	Kmat1	KMAT	Sales order
20	Kmat2	KMAT	Sales order



: : 80	Fert1	FERT	→ No setting in profile, material is not configurable
--------------	-------	------	---

Items 10 and 20 both represent separate configurations. The structure information is in SD only.

For this reason, object variable \$ROOT must not be used if you use dependencies for item 80 (Fert1), because in this example item 20 (Kmat2) is the parent or root of item 80 (Fert1).

**See also:**

[Profile Scenarios: Possible Combinations \[Page 63\]](#)

---

Process: Order BOM

## Process: Order BOM

### Purpose

You use this processing option if you want to make customer-specific changes to the BOM of a material that you configure in the sales order. In the sales order, you assign values to the characteristics of the header material, but the BOM is not exploded in the sales order.

The BOM components that match these values are not determined until you call a special processing function. Subordinate configurable materials can be configured in this function, and you can make order-specific changes to the BOM.

There are 2 types of order BOMs:

- The **result-oriented order BOM** saves the configured BOM with the manual changes.
- The **knowledge-based order BOM** saves the super BOM with all manual changes and dependencies, not the configured BOM. When you explode the BOM, the dependencies are processed dynamically and only the selected items are displayed.



If you change the characteristic values assigned to the material in transaction VA01, VA02, or CU51, this may affect the order BOM.

### Prerequisites

The configuration profile of the configurable material has the *Process: Order BOM* indicator selected, with a single-level or multi-level BOM explosion.

### Process Flow

#### Maintenance of Order BOM not Allowed in Sales Order

1. Create a sales order and enter the configurable material as an order item.
2. You see the configuration editor, where you assign values to the characteristics of the header material. As you enter characteristic values, dependencies for the characteristics and values are processed.

The BOM is not exploded in the sales order. You cannot assign values to subordinate configurable materials in the sales order.

The header material appears as a sales order item.

3. Once you have saved the sales order, you can process the BOM for the sales order by choosing *Bill of material* → *Order BOM* → *Maintain multi-level* from the bills of material menu. The values assigned to the material in the sales order are displayed and cannot be changed.

#### Maintenance of Order BOM Allowed in Sales Order

You can only use this function if you select *Result-oriented BOM* in the configuration profile first.

1. Create a sales order and enter the configurable material as an order item.

**Process: Order BOM**

2. You see the configuration editor, where you assign values to the characteristics of the material. As you enter characteristic values, dependencies for the characteristics and values are processed.
3. Choose *Engineering* to process the order BOM.
  - You can assign values to subordinate configurable materials or change values already assigned.
  - Exit configuration and return to the sales order.
 

The order BOM is saved as a result-oriented order BOM when you save the sales order.
4. The header material is shown in the sales order as an order item.

## Result

In material requirements planning (MRP), requirements are determined for the order item. To ensure that MRP checks whether a material has an order BOM, set the *Sales order BOM* indicator in Customizing by choosing *Planning → BOM Explosion/Determine Routing → Define BOM explosion control*.

### See also:

For more information on maintaining order BOMs with and without Variant Configuration, together with the preceding and subsequent process, see the *SAP Library* under *Logistics → PP Production Planning → PP Order BOM*.

The following sections deal with maintenance of order BOMs with variant configuration:

[Order BOMs for Variant Configuration \[Ext.\]](#)

[Settings in the Configuration Profile \[Ext.\]](#)

[Creating a Knowledge-Based Order BOM \[Ext.\]](#)

[Creating a Result-Oriented Order BOM \[Ext.\]](#)

[Instantiating Configurable Materials \[Ext.\]](#)

[Fixing Configurable Assemblies \[Ext.\]](#)

## Controlling the BOM Explosion

# Controlling the BOM Explosion

The following factors determine how a BOM is exploded:

- BOM usage
- BOM application
- Filter defined in the configuration profile

## BOM Usage

When you create a BOM, you must enter a usage. This usage defines which application areas (for example, sales and distribution, engineering/design, or production) a BOM can be exploded in. For example, the standard R/3 System supports usage 1 for BOM explosion in production. A BOM with usage 3 is relevant to sales and distribution, too.

The usage determines which indicators are active in the *Item status* dataset of BOM maintenance functions. This is where you specify whether BOM items are relevant to Sales or Production, for example.

## Features

The BOM explosion is influenced by the application entered in the configuration profile. The application determines which usages of a BOM can be exploded and in which order of priority.

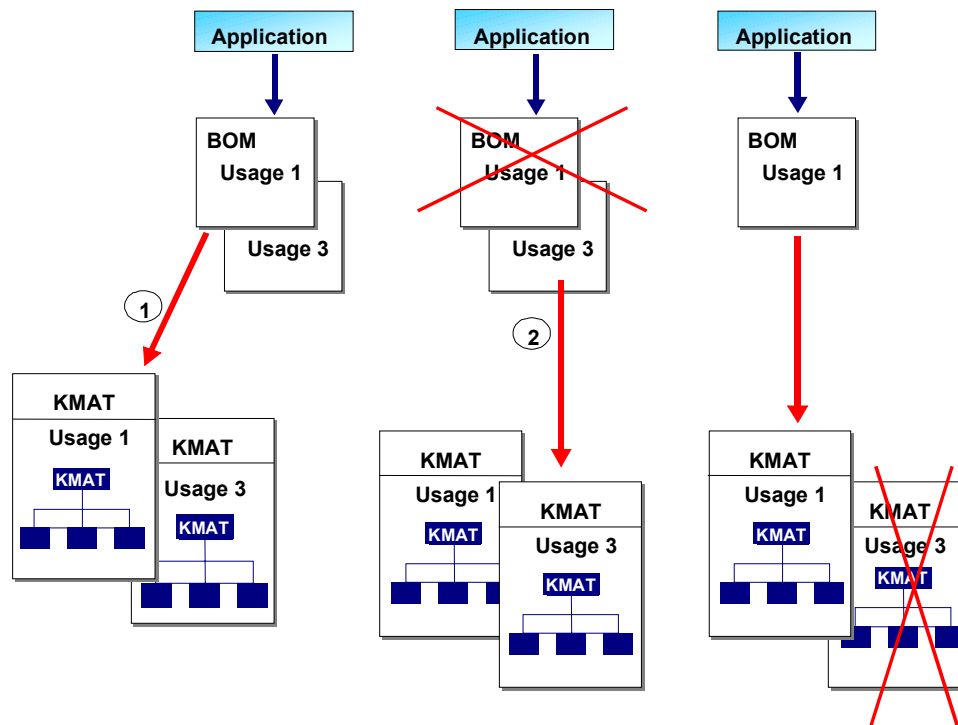
You enter the application for exploding the BOM in the configuration profile. However, please note that you can also enter an application for a sales order item category. This entry overwrites the setting in the configuration profile. Check the settings for the item category in Customizing for Sales and Distribution, by choosing *Sales* → *Sales Documents* → *Sales Document Items* → *Define item categories*.

If you select the *Sales order* process in the configuration profile, ensure that the application supports explosion of sales-relevant BOMs and that sales-relevant BOMs have the highest priority.



You define an application that explodes first usage 1 and then usage 3. When you explode the BOM, the system first checks whether a BOM with usage 1 exists. If a BOM with usage 1 does exist, the BOM is exploded. If no BOM with usage 1 exists, the system checks whether a BOM of usage 3 exists.

## Controlling the BOM Explosion



### Handling in the Sales Order

- If usage 1 is supported by the application, the BOM for the material is found but cannot be exploded, because usage 1 in the standard R/3 System is not relevant to sales and distribution.
- If usage 1 is **not** supported by the application, the system does not recognize in the sales order that a BOM exists for the material.

### Filters in the Configuration Profile

In the configuration profile, you can define filters so that only certain items are displayed (see [Filters for BOM Explosion \[Page 34\]](#)).

## No BOM Explosion

## No BOM Explosion

### Use

You want to configure a material that can be supplied in a range of variants.

The material has a BOM that is not relevant to the sales order. The customer requires only the characteristics that describe the variant. The components required are not determined until the MRP or production stage, so it is not necessary to explode the BOM in the sales order.

If the BOM contains other configurable materials, you cannot configure these materials .



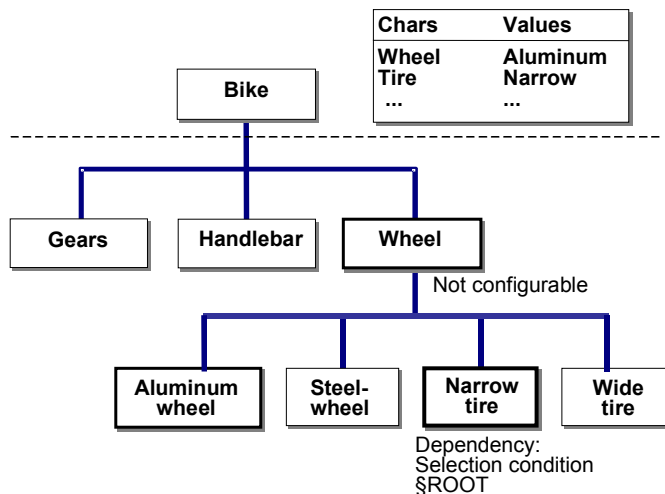
#### Simple Example

The material is a bike that the customers can put together according to their requirements. One customer wants a red racing bike with narrow tires, 21 gears, and a racing handlebar. Another customer may want a green trekking bike with wide tires, a trekking handlebar, and 12 gears. The bill of material for the bike includes the various components that can make up the variants of the bike.

You enter the features of material **Bike** in the form of characteristic values. The characteristic values assigned to the bike are saved in the sales order. The sales order is passed on to the shop floor. On the shop floor, the characteristic values assigned in the sales order are used to determine which parts are required to build the bike. If the customer ordered a bike with narrow tires and 21 gears, the production BOM contains exactly these components.

#### Example with Several Configurable BOM Items

The materials BIKE and WHEEL are configurable. The wheel of a bike can have its own configuration profile. This is especially useful if you also want to sell wheels separately. However, when you use the wheel as a component of a bike, you cannot configure the wheel separately. If the wheel has its own characteristics, you must define object dependencies that set the characteristic values for the wheel, according to the characteristic values assigned to the bike.



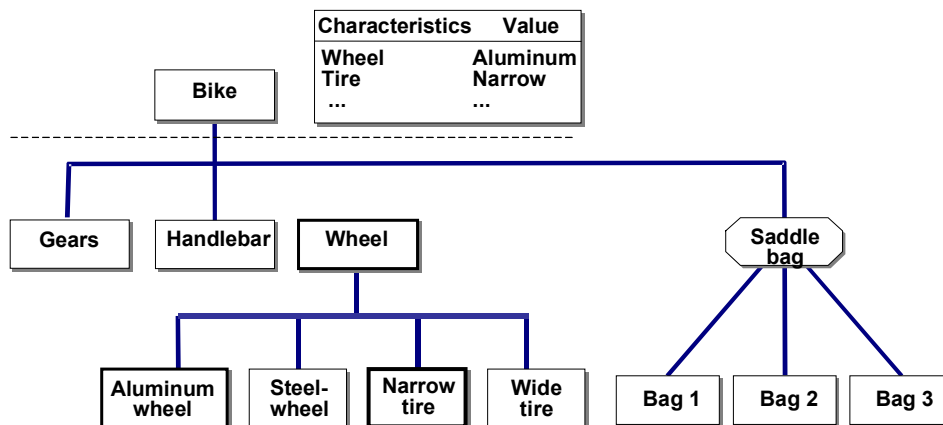
If you always want to sell the wheel as part of a finished bike, not as a separate product, you do not need a separate configuration profile for the wheel. You only need to describe the wheels

## No BOM Explosion

when you configure the bike as a whole. To enable you do this, the characteristics of the wheels must be assigned to the configurable bike. By assigning characteristic values to the bike, you ensure that the necessary components and operations for the wheel are selected.

## Example with a Class as a BOM Item

In the BOM for the bike, you can enter a class as a BOM item. For example, you can classify different saddlebags for a bike in a class called **Saddlebags**. To do this, use the classification functions in the classification system (see the SAP Library, under *CA Classification*).



In the BOM for the bike, you enter class **Saddlebags** as a class item. You do not enter each individual saddlebag. When you configure the bike in a sales order, the class item is replaced by a component. For more information on classes as BOM items, see [Selecting BOM Items \[Page 14\]](#).

In this case, too, characteristics are used to describe the different bags, so that the BOM need not be exploded in the sales order.

## Pricing

In pricing, the system reads the conditions you maintained for the header material. For example, the price of a material can be determined using pricing condition PR00. You maintain pricing condition PR00 with the sales master data function *Conditions* → *Pricing* → *Material price* → *Create*. You can also maintain a customer-specific or price list-specific price for a material.



PR00 is only a mandatory condition in the standard pricing procedure. Depending on the Customizing modifications in your company, PR00 may no longer be a mandatory condition.

Individual prices for BOM components are not included in the calculation. However, you can define surcharges and discounts that are dependent on the characteristic values assigned, in the form of variant conditions. You can use dependencies to determine when these conditions are active.

## No BOM Explosion

### Prerequisites

- Materials whose BOMs are not to be exploded in the sales order have the following settings in their configuration profile:
  - Configuration parameter: *BOM explosion: None*
  - Under *UserInterf* in the configuration profile, you cannot specify the screens allowed for configuration.
- The configurable material must have all the characteristics that are needed to select the components required.
- LOW-LEVEL configuration principles are used for exploding BOMs and task lists because the BOM components and task list operations are first reported during requirements planning or production (see also: [Low-Level Configuration \[Page 280\]](#)).
  - You can only use constraints when assigning characteristic values to the header material. However, constraints do not affect the BOM explosion in MRP. For this reason, you cannot use constraints to infer values for subordinate configurable materials.  
For more information on constraints, see [Constraints \[Page 106\]](#).
  - This is a single-level configuration, so the dependencies for configuration profiles of subordinate configurable materials are ignored when you explode the BOM. Dependencies for the BOM items and operations of subordinate assemblies are processed (except for constraints).

### Process Flow

- Create a sales order and enter the configurable material as an order item.
- You see the configuration editor, where you assign values to the characteristics of the material. As you enter characteristic values, dependencies for the characteristics and values are processed.  
You cannot go to other overviews in the configuration editor if you select *None* under *BOM explosion*.
- In MRP, requirements are determined for the item in the sales order. In the planned order or production order, the components of the BOM and the operations of the task list are determined according to the characteristic values assigned in the sales order. Low-level configuration applies.



## Single-Level BOM Explosion

### Purpose

The BOM is exploded on one level on the result screen in the sales order. Other configurable material can be contained in the BOM and these can be configured. However, the BOMs of these materials are not exploded in the sales order.

You can combine this setting with *Process: Sales order* when dealing with configurable material of the BOM that you want to assemble in a set structure.

See also: [Process: Sales Order \[Page 37\]](#)

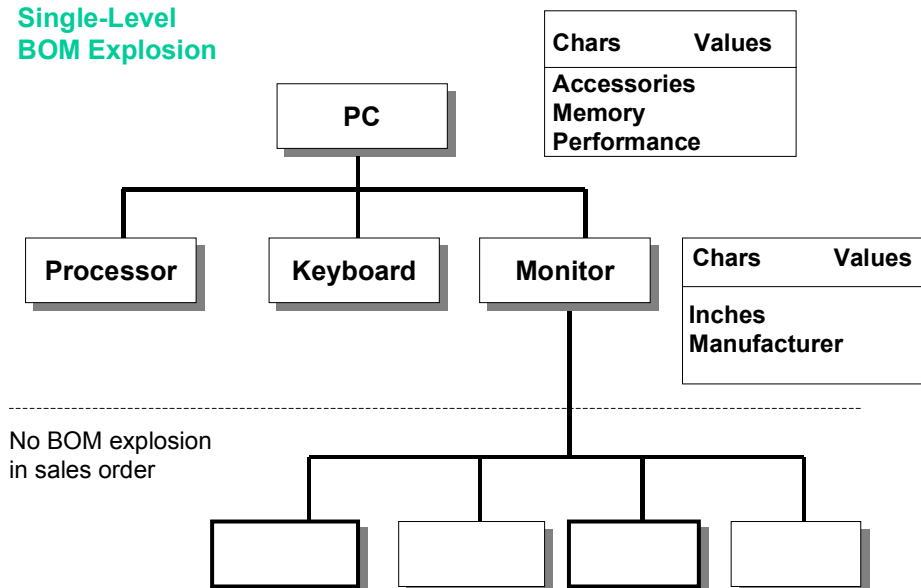


### Example

The materials PC and MONITOR are configurable. The monitor has its own configuration profile and characteristics. The configurable PC has the setting *BOM explosion: single level* in its configuration profile. On the result screen in the sales order, you see the selected components.

Configurable material MONITOR can also be configured in the sales order. However, the BOM of the PC is not exploded, because a single-level explosion only is defined for the header material. The configuration parameters for the material MONITOR are not important in this instance.

#### Single-Level BOM Explosion



### Features

- All configurable items are configured in the sales order, not only sales-relevant items. The BOMs of these materials are not exploded in the sales order.

## Single-Level BOM Explosion



You can use filters in the configuration profile to restrict the scope of the BOM items (see [Filters for BOM Explosion \[Page 34\]](#)).

- Sales-relevant items are copied to the sales order as order items.
- The characteristic values assigned to each configured item are saved.
- In MRP the required components are determined based on the characteristic value assignments for the head material and the subordinate items as well as the task list operations based on low-level configuration. (See also [Low Level Configuration \[Page 280\]](#)).
- If you define selection conditions for the BOM items of a subordinate configurable assembly, and these selection conditions refer to the characteristics of the assembly, you must use the object variable \$PARENT to refer to the characteristics.



The selection conditions for BOM components of the monitor must refer to the characteristics of the monitor with object variable \$PARENT – for example, \$PARENT.MANUFACTURER = 'Sony'.

- This type of order processing lets you use constraints to infer values for configurable materials.

## Prerequisites

The configuration profile of the header material has the configuration parameter *BOM explosion: single level*.

## Process Flow

1. Create a sales order and enter the configurable material as an order item.
2. You see the configuration editor, where you assign values to the characteristics of the material. You can then display the result of the BOM explosion by choosing *Result*.
3. If any of the selected BOM components are also configurable materials, you can assign values to the characteristics of these materials. However, the BOMs of these materials are not exploded.
4. The header material appears as a sales order item. If the BOM contains further sales-relevant items, these are included as sub-items. The characteristic values assigned to the header material and the subordinate configurable materials are saved.
5. In the planned order or production order, BOM components and operations for the configured materials are determined according to the characteristic values assigned.

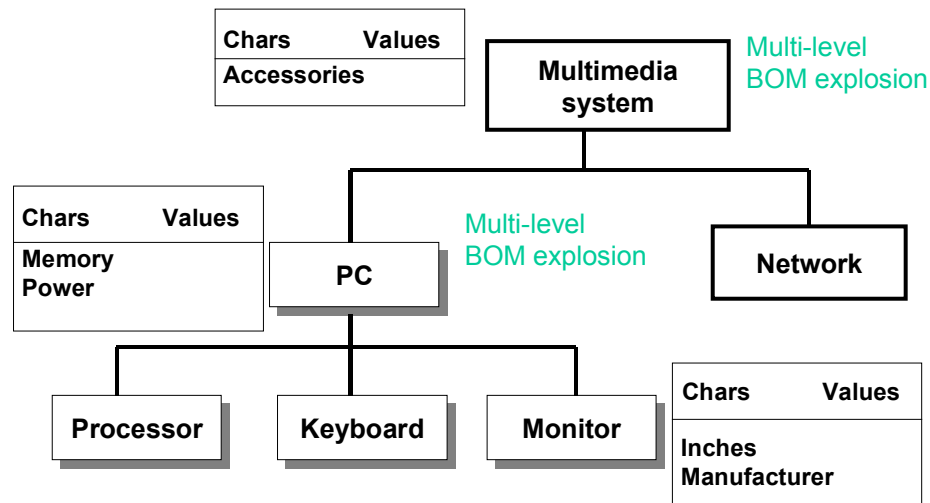
## Multi-Level BOM Explosion

### Purpose

The setting *BOM explosion: multi-level* means that multiple levels of the BOM of a configurable material are exploded in the sales order, if the configuration profiles of the configurable assemblies also support BOM explosion. You can configure assemblies across multiple levels.



Variant product MULTIMEDIA\_WORKSTATION supports multi-level BOM explosion. The BOM contains configurable material PC. This material supports single-level BOM explosion. The BOM of the PC contains another configurable material MONITOR.



### Features

- All configurable items are configured in the sales order, not only sales-relevant items.



You can use filters in the configuration profile to restrict the scope of the BOM items (see [Filters for BOM Explosion \[Page 34\]](#)).

- Sales-relevant items are copied to the sales order as order items.
- The characteristic values assigned to each configured item are saved.

### Multi-Level BOM Explosion

- In MRP, components and operations selected depend on the characteristic values assigned according to the principles of low-level configuration (see [Low-Level Configuration \[Page 280\]](#)).
- If you define selection conditions for the BOM items of a subordinate configurable assembly, and these selection conditions refer to the characteristics of the assembly, you must use the object variable \$PARENT to refer to the characteristics.



The selection conditions for BOM components of the PC must refer to the characteristics of the monitor with object variable \$PARENT – for example, `$PARENT.MANUFACTURER = 'Sony'`.

- This type of order processing lets you use constraints to infer values for configurable materials on different levels.

### Prerequisites

The configuration profile of the header material has the configuration parameter *BOM explosion: multi-level*.

The subordinate materials whose BOMs are to be exploded in the sales order must also have a BOM explosion parameter (single-level or, preferably, multi-level).

### Process Flow

1. Create a sales order and enter the configurable material as an order item.
2. You see the configuration editor, where you assign values to the characteristics of the header material. You can then display the result of the BOM explosion by choosing *Result*.
3. If any of the selected BOM components are also configurable materials, you can assign values to the characteristics of these materials. Depending on the configuration parameters, the BOMs of these materials are exploded on the result screen.
4. The header material appears as a sales order item. If the BOM contains further sales-relevant items, these are included as sub-items. The characteristic values assigned to the header material and the subordinate configurable materials are saved.
5. In the planned order or production order, BOM components and operations for the configured materials are determined according to the characteristic values assigned.

## Interface Settings

### Use

If you want to group and order characteristics on the value assignment screen according to your own criteria (*Values* → *Interface design*), you must assign a name to the interface design on the *Interface* tab of the configuration profile. This activates the function on the value assignment screen.

The interface design you define on the value assignment screen is saved for the configurable object. If another configurable object is assigned to the same class, you can enter the interface design for this object, too. The characteristics are then displayed as you defined in the interface design.

However, you can only use an interface design for objects that are assigned to the same class. If the class assignment is not identical (for example, in multiple classification), you cannot enter the interface design you defined.

You can choose *Settings* to define object-specific settings for functions in the configuration editor. These settings are defaults for configuration, and can be overwritten for your user in the configuration editor.

### Prerequisites

The configurable object must be assigned to a class.

#### See also:

[Defining an Interface Design \[Page 248\]](#)

[Defining Settings for the Language \[Page 54\]](#)

[Defining the Scope and Display Options for Characteristics \[Page 55\]](#)

[Defining Settings for Pricing \[Page 57\]](#)

[Defining Settings for the Default Values \[Page 58\]](#)

[Defining Settings for the Configurator \[Page 59\]](#)

[Settings for Variant Matching \[Page 60\]](#)

## Defining Settings for the Language

# Defining Settings for the Language

## Use

In the standard system, characteristics and characteristic values in character format (CHAR) are shown on the characteristic value assignment screen with their language-dependent descriptions in the logon language. If a characteristic does not have a description in your logon language, you see the language-independent name of the characteristic.

You can define the following settings for the language:

- You can specify whether characteristics and values are displayed with their language-dependent descriptions or their language-independent names.
- You can specify whether the settings apply to characteristics only, values only, or both characteristics and values.
- You can specify an alternative language for the description, in case there is no description in the language you select.

For example, you can define French as the alternative language. If you are logged on in English and characteristics and values do not have a description in English, you see the descriptions in French.

## Procedure

You can define these settings in the configuration profile and on the characteristic value assignment screen.

1. In the configuration profile, choose *Settings*, and on the characteristic value assignment screen choose *View* → *Settings*.
2. Choose the *Language* tab.
  - You can specify the language for displaying descriptions.
  - You can only specify an alternative language in the configuration profile. This language is shown on the value assignment screen.
3. If you do not use the default values, you can save user-specific settings.



If you do not define user-specific settings, or if you delete the user-specific settings, the settings in the configuration profile apply.

### For Settings on the Characteristic Value Assignment Screen Only:

4. If the user-specific settings are different to the configuration profile settings and the current settings, you can display the settings next to each other by choosing *Comparison*.

The current settings only apply to the transaction you are currently processing.

## Defining the Scope and Display Options for Characteristics

### Use

You can define whether characteristic values are displayed on the value assignment screen.

- In the standard system, the settings defined in characteristics maintenance apply.
- You can define that no characteristic values are displayed on the value assignment screen, regardless of the settings in characteristics maintenance.
- You can define that values for all characteristics are displayed on the value assignment screen, regardless of the settings in characteristics maintenance.

You can also define whether each characteristic is shown on a separate screen, or whether characteristics are listed beneath each other.

You can hide and show characteristics on the interface.

- You can display required characteristics only or optional characteristics only.
- You can display characteristics with assigned values only, or characteristics without assigned values only.

In the standard system, all characteristics are displayed.

- In the standard system, characteristics and values that are not allowed due to dependencies are hidden. To show these characteristics and values, select *With excluded chars*. You cannot assign values to these characteristics or select these values.

If you define that all values are shown on the value assignment screen, you cannot show excluded values.

### Procedure

You can define these settings in the configuration profile and on the characteristic value assignment screen.

3. In the configuration profile, choose *Settings*, and on the characteristic value assignment screen choose *View* → *Settings*.

You see a dialog box.

4. Choose the *Display options* or *Scope* tab.
5. If you do not use the default values, you can save user-specific settings.



If you do not define user-specific settings, or if you delete the user-specific settings, the settings in the configuration profile apply.

#### For Settings on the Characteristic Value Assignment Screen Only:

6. If the user-specific settings are different to the configuration profile settings and the current settings, you can display the settings next to each other by choosing *Comparison*.

---

**Defining the Scope and Display Options for Characteristics**

The current settings only apply to the transaction you are currently processing.



## Defining Settings for Pricing

### Use

You can define that the price of a variant is displayed on the characteristic value assignment screen permanently or only on request. The price of a variant product depends on the characteristics selected and can be influenced by variant conditions.

You can have the price displayed:

- Permanently
- On request only

Permanent pricing slows down system performance.

If you only display pricing on request, the *Pricing* pushbutton is shown on the characteristic value assignment screen, where you can choose it to start pricing.

### Procedure

You can define these settings in the configuration profile and on the characteristic value assignment screen.

7. In the configuration profile, choose *Settings*, and on the characteristic value assignment screen choose *View* → *Settings*.

You see a dialog box.

8. Choose the *Pricing* tab.
9. If you do not use the default values, you can save user-specific settings.



If you do not define user-specific settings, or if you delete the user-specific settings, the settings in the configuration profile apply.

#### **For Settings on the Characteristic Value Assignment Screen Only:**

10. If the user-specific settings are different to the configuration profile settings and the current settings, you can display the settings next to each other by choosing *Comparison*.

The current settings only apply to the transaction you are currently processing.

## Defining Settings for Default Values

## Defining Settings for Default Values

### Use

You can define how the default settings in characteristics maintenance are interpreted on the value assignment screen:

- The standard setting interprets default values as user entries. The characteristic is interpreted as having an assigned value.
- Default values are display **only**, and must be explicitly set by the user. If the default value is not confirmed, the characteristic is interpreted as not having an assigned value.

### Procedure

You can define these settings in the configuration profile and on the characteristic value assignment screen.

11. In the configuration profile, choose *Settings*, and on the characteristic value assignment screen choose *View* → *Settings*.

You see a dialog box.

12. Choose the *Default values* tab.

- To have default values interpreted as user entries, select *Enter* and *Copy*.
- To have default values confirmed by the user, select *Confirm*.

3. If you do not use the default values, you can save user-specific settings.



If you do not define user-specific settings, or if you delete the user-specific settings, the settings in the configuration profile apply.

#### **For Settings on the Characteristic Value Assignment Screen Only:**

4. If the user-specific settings are different to the configuration profile settings and the current settings, you can display the settings next to each other by choosing *Comparison*.

The current settings only apply to the transaction you are currently processing.

## Defining Settings for the Configurator

### Use

On the characteristic value assignment screen, you can deactivate processing of dependencies. This considerably reduces the response time when you select a value and confirm, because no dependencies are read. However, there is no check as to whether the values you assign are consistent.

When you reactivate the configurator, all dependencies are processed at once. Any inconsistencies are collected and displayed. This may make it difficult to trace the reason for an inconsistency.

### Procedure

You can define these settings in the configuration profile and on the characteristic value assignment screen.

13. In the configuration profile, choose *Settings*, and on the characteristic value assignment screen choose *View* → *Settings*.

You see a dialog box.

14. Choose the *Configurator* tab.

15. If you do not use the default values, you can save user-specific settings.



If you do not define user-specific settings, or if you delete the user-specific settings, the settings in the configuration profile apply.

#### **For Settings on the Characteristic Value Assignment Screen Only:**

16. If the user-specific settings are different to the configuration profile settings and the current settings, you can display the settings next to each other by choosing *Comparison*.

The current settings only apply to the transaction you are currently processing.

## Settings for Variant Matching

## Settings for Variant Matching

### Use

When you configure a material, you can check whether a material variant already exists that has the same characteristic values.

You can display existing material variants on the characteristic value assignment screen. However, the configurable material is not automatically replaced by the variant. The displayed variants are for information only.

You can specify the following for variant matching on the characteristic value assignment screen:

- The search is started only on request.  
If you do this, you see a pushbutton on the characteristic value assignment screen, which you select to display the material variants.
- The search is on permanently.  
As soon as a material variant with the same characteristic values is found, you see the material number on the characteristic value assignment screen.

You can define a strategy as to whether a material that matches your entries is found after a partial configuration or only when the configuration is complete.

- If you select the *Partial confign* indicator, a material is displayed as soon as the characteristic values you assign partly match the values assigned to a material variant. The characteristic values you assign must be contained in the material variant, but the material variant can also contain other characteristics. You can select the material variant and display its characteristic values.



If you want to replace the configurable material in the sales order with the material variant, the values assigned to the material variant and the values assigned to the configurable material must be identical. Assign values to the remaining characteristics of the configurable material so that they match the values of the material variant.

- If you select the *Complete configuration* indicator, a material is only displayed if the characteristic value assignment process is complete and all characteristic values assigned to the configurable material match the values assigned to a material variant. If this happens, the configurable material can be replaced in the sales order with the material variant.
- If you select *Chars without values*, material variants are found if they do not have a value for one or more of the characteristics in the configuration.

If this indicator is not set, only material variants that have values for all the characteristics that have values in the configuration are found.

### Procedure

You can define the settings for variant matching in the configuration profile and on the characteristic value assignment screen.

## Settings for Variant Matching

17. In the configuration profile, choose *Settings*, and on the characteristic value assignment screen choose *View* → *Settings*.

You see a dialog box.

18. Choose the *Variant matching* tab.

19. If you do not use the default values, you can save user-specific settings.



If you do not define user-specific settings, or if you delete the user-specific settings, the settings in the configuration profile apply.

### **For Settings on the Characteristic Value Assignment Screen Only:**

20. If the user-specific settings are different to the configuration profile settings and the current settings, you can display the settings next to each other by choosing *Comparison*.

The current settings only apply to the transaction you are currently processing.

See also:

[Variant Matching in the Sales Order \[Page 273\]](#)

## Changing/Displaying/Deleting Configuration Profiles

## Changing/Displaying/Deleting Configuration Profiles

### Procedure

#### Change Configuration Profile

1. From the variant configuration menu, choose *Configuration Profile* → *Change*.

You see a dialog box for specifying the object whose configuration profile you want to change. Select an object and confirm your entry.




To switch to another object, choose *Extras* → *Change confble obj*.

2. Enter the name of the object.
3. Choose *Profile overview*.

You see the overview of profiles created for the object.

You can create new profiles for the configurable object by choosing *Edit* → *New entries*.

4. Choose  *Profile detail* to see the detail screen. You can change the settings for the profile.



You cannot change the settings *Process: Sales order (SET)* and *Order BOM* for a material once the material has been configured with the profile. If only the header material has been changed, the settings can be changed, but not if subordinate items of the BOM have been changed.

#### Display Configuration Profile


1. From the variant configuration menu, choose *Configuration profile* → *Display*.
2. For the rest of the process, see *Changing a Configuration Profile* above. However, you cannot make changes in display mode.

#### Deleting a Configuration Profile

You can only delete an existing profile by using *Configuration profile* → *Change* if the profile has not been used to configure the object. If a profile has been used to configure a material in a sales order, for example, you can no longer delete the profile. The profile can only be deactivated by changing the status.

You can only use engineering change management to delete configuration profiles if no configured objects exist, or if the date of the configuration of all configured objects is before the deletion date.

#### Renaming a Configuration Profile

You can change the name of a configuration profile on the profile overview by choosing  *Rename*.

Possible Combinations of Configuration Profiles

## Possible Combinations of Configuration Profiles

You often need to work with several configuration profiles within one configuration.

For example, the materials PC, MONITOR and KEYBOARD are configurable. The monitor and keyboard are also sold separately. You need to create separate configuration profiles for them, because this involves a multi-level configuration.

However, there are restrictions on the profiles you can select for subordinate components. You need to work with profiles that are allowed in combination with the scenario for the header material.

The following table shows the possible combinations:

Header Material	Subordinate Materials	Plnd/Prod. No BOM explosion	Plnd/Prod. Single-/multi-level BOM explosion	Single-/multi-level sales order	Single- /multi-level order BOM - knowledge-based -	Single- /multi-level order BOM - result-oriented -
Plnd/Prod. Order No BOM explosion		C	C	A	C	C
Plnd/Prod. Order Single-/multi-level BOM explosion		C	A	C	C	C
Single-/multi-level sales order		C	C	A	C	C
Single-/multi-level order BOM - knowledge-based -		C	C	A	A	C
Single-/multi-level order BOM - result-oriented -		C	C	A	C	A

Abbreviations:

A = Possible/allowed

B = Not allowed

C = No effect:

- The material is handled as though it had no configuration profile.
- Profile selection shows no options.

### Comment

The only difference between the scenarios "single-level" and "multi-level" is that the "single-level" setting only explodes the assembly of the header material (single-level explosion). You can only assign values to the components of the header material.

---

**Possible Combinations of Configuration Profiles****Note**

You can run check report RCU\_CHECK\_SUB\_PROF\_NO\_BOM\_EXPL to check configuration profiles on subordinate levels that you have defined with *No BOM explosion*, and change them if required.



## Dependencies

### Purpose

Dependencies let you do the following:

- Describe the interdependencies between characteristics and characteristic values
- Control which components are selected from a bill of material (BOM) and which operations are selected from a task list
- Change the values of fields in BOM items and operations during configuration

You use a special syntax in the dependency editor to define dependencies.

### Dependency Types

The SAP System supports the following types of dependencies:

- Preconditions
- Selection conditions
- Actions (obsolete)
- Procedures
- Constraints

### Global and Local Dependencies

The differences between global and local dependencies are as follows:

- Global dependencies are created centrally and can be assigned to several objects.
- Local dependencies are created for one object and can only be used with this object.

### Integration

You can use dependencies in the following components:

- Component CA – Classification System
- Component LO – Variant Configuration

---

Global Object Dependencies

## Global Object Dependencies

### Use

Global dependencies have the following properties:

- They are independent of any object.
- They are identified by a name that you assign, and are maintained and managed centrally.
- If you change a global dependency, the change affects all the objects where the dependency is used.

### Integration

There are special allocation functions that enable you to allocate global dependencies to individual objects.

The documentation of the applications tells you how to use these allocation functions.

#### See also:

R/3 Library:

*Characteristics (CA-CL-CHR)*

*Bills of Material (PP-BD-BOM)*

*PP Routings*

*PP–PI Master recipes*

## Creating Global Dependencies

1. From the variant configuration menu, choose *Dependency* → *Single dependency* → *Create*.
2. You see the initial screen. Enter a name for your dependency.  
If you want to create your dependency as of a specific date, enter a change number.



To copy an existing dependency, choose *Copy from*. The basic data and source code of the existing dependency you enter are copied to the new dependency.

Confirm your entry.

3. You see the basic data screen.
  - On the basic data screen, enter a language-dependent description for the dependency.
  - To enter descriptions in different languages, choose *Descriptions*.
  - To enter a long texts for the dependency, choose *Extras* → *Documentation*.
  - In the standard R/3 System, you see the status '*In preparation*' when you first create a dependency. You cannot set the status to '*Released*' until you have written syntactically correct source code.



You can check the possible statuses in Customizing for *Variant Configuration*.

4. If you select the *Dependency editor* pushbutton, you see the editor, where you enter the source code for the dependency.

Once you have entered the source code, choose *Check* to check whether your source code contains errors.

- The system checks whether the dependency syntax is correct.
- For alphanumeric characteristics, the system checks whether the values you enter are correct.



The system does not check the values of numeric characteristics.

5. Save your source code.  
If you want to save source code that contains errors, choose *Save without generating*. The system sets the status of the dependency to '*Locked*'. You cannot release the dependency until the source code is error-free.
6. You see the basic data screen again.  
If the source code contains no errors and you want to use the dependency, change the status to '*Released*'.  
Save your dependency.

## Changing Global Dependencies

## Changing Global Dependencies

### Procedure

1. From the variant configuration menu, choose *Dependency* → *Single dependency* → *Change*.

If you want to make your change on a specific date, enter a valid change number.

2. You can change the basic data and the source code of the dependency.
3. Save your changes.



You can only change global dependencies that have been used several times centrally.

The changes you make to the dependency affect all objects where the dependency is used.

## Displaying Global Dependencies

1. From the variant configuration menu, choose *Dependency* → *Single dependency* → *Display*.

You can display a dependency on today's date, or on a specific date if the dependency has been processed with a change number. If you want to display a dependency on a specific date, enter the valid-from date you require.

Confirm your entries.

2. You see the basic data screen.
3. To display the source code, choose *Goto* → *Dependency editor*.

---

**Local Object Dependencies**

## Local Object Dependencies

### Use

Local dependencies have the following properties:

- They are only available to the object for which you create them
- You cannot use the central maintenance functions on these dependencies, and you cannot allocate them to other objects
- They are identified by a number assigned by the system, not an external name

Only use local dependencies if you are certain that the dependency will not be needed elsewhere.

### Integration

You create local dependencies in the application for the object. For more information, refer to the document on the application.

#### See also:

R/3 Library:

*Characteristics (CA-CL-CHR)*

*Bills of Material (PP-BD-BOM)*

*PP Routings*

*PP-PI Master recipes*

## Maintenance Authorizations for Dependencies

### Use

You can enter a maintenance authorization on the basic data screen for creating a single dependency, a dependency net, or a constraint. The maintenance authorization fulfills two functions.

### Maintenance Authorization for Allocating Dependencies to Objects

In Customizing for *Variant Configuration*, under *Define Maintenance Authorizations*, the maintenance authorizations that are predefined in the standard system only allow dependencies to be allocated to the following objects:

- BOMs
- Characteristics and characteristic values
- Task lists

### Maintenance Authorization for Maintaining Dependencies

In Customizing for *Variant Configuration*, you can define new maintenance authorizations by choosing *Define Maintenance Authorizations*. You can assign these authorizations to dependencies.

In the user master record of the relevant users, you enter authorization object C\_LOVC\_DEP with the maintenance authorization value that you want the user to have.

Each user can only maintain dependencies whose maintenance authorization matches that in their user master record.



A dependency has maintenance authorization 100.

User A has the authorization to maintain dependencies with maintenance authorization 050–150.

User B has the authorization to maintain dependencies with maintenance authorization 200–300.

Only user A can maintain the dependency with maintenance authorization 100.

When you restrict a dependency to certain user groups, please note the following:

- If a dependency net or a dependency has a maintenance authorization, the system checks whether a user is allowed to use this maintenance authorization. The system checks activity 02 (change).
- If a user changes, deletes, or regenerates a dependency net that has a maintenance authorization, the system checks whether the user has the authorization to do so.
- Authorization to allocate a dependency to an object is defined in the object.

If you delete a local dependency, you delete the entire dependency, so the system checks your authorization to change or delete a dependency, as well as your authorization for the object, if you do this.

**Maintenance Authorizations for Dependencies**

- If you copy a dependency that has a maintenance authorization, the system checks whether you have this authorization in your user master record. If you do not, the authorization group is not copied to the new dependency.



## Preconditions

### Use

You can use preconditions to hide characteristics and characteristic values that are not allowed and thereby ensure that the configuration of an object is consistent.

You can allocate preconditions to the following objects:

- A characteristic that you want to hide
- A characteristic value that you want to hide

In the precondition, you define the circumstances under which a characteristic or value is hidden.



You can also use restrictable characteristics to restrict the allowed values of characteristics when you configure and object.

### Features

A precondition is fulfilled if the condition you enter is either true or not violated.



A precondition is fulfilled if:

- a. The specified value is selected for the specified characteristic
- b. **No** value is selected for the specified characteristic

The precondition is not fulfilled if a different value is selected for the specified characteristic.

**Example: Precondition for a Characteristic Value**

## Example: Precondition for a Characteristic Value

You have a configurable material, BIKE. The material has characteristics MODEL and GEARS with the following values:

Characteristic	Values	Condition
MODEL	Racing	
	Standard	
	Mountain	
	Tandem	
GEARS	10	
	12	
	17	
	21	MODEL = 'Racing'

Only a racing bicycle can have 21 gears.

### Procedure

1. Create a precondition.
2. This precondition has the following source code:  

```
MODEL EQ 'Racing'
```
3. Allocate the precondition to value 21 of characteristic GEARS, because this value is affected by the precondition.

### Result

The system checks whether the value 'Racing' is set for characteristic MODEL.

- If characteristic MODEL has another value, you do not see the value 21 for characteristic GEARS.
- If characteristic MODEL has the value 'Racing', you see value 21 for GEARS.
- If characteristic MODEL does not have a value, you see characteristic value 21 for characteristic GEARS, because the precondition is not violated.

### Completing the Condition

You want value 21 hidden for characteristic GEARS if characteristic MODEL does not have a value. Add the following condition to the source code:

```
MODEL eq 'Racing' and Specified MODEL
```

There are now two conditions to fulfill for this precondition:

1. Characteristic MODEL must have a value.

---

**Example: Precondition for a Characteristic Value**

2. Characteristic MODEL must have the value 'Racing'.

---

**Precondition for a Characteristic**

## Precondition for a Characteristic

You have a configurable material, BIKE. This material has characteristic MODEL.

Characteristic	Values	Condition
MODEL	Racing	
	Standard	
	Mountain	
	Tandem	
TANDEM_SADDLE		MODEL = 'Tandem'

If the value 'Tandem' is selected for MODEL, you see characteristic TANDEM\_SADDLE, which was hidden before.

## Procedure

1. Create a precondition.
2. This precondition has the following source code:

```
MODEL eq 'Tandem'
```

3. Allocate the precondition to characteristic TANDEM\_SADDLE, because this characteristic is affected by the precondition.

## Result

The system checks whether the value 'Tandem' is set for characteristic MODEL.

- If it is, you see characteristic TANDEM\_SADDLE.
- If characteristic MODEL has another value, you do not see characteristic TANDEM\_SADDLE.
- If characteristic MODEL does not have a value, you see characteristic TANDEM\_SADDLE, because a lack of value does not violate the precondition.

## Completing the Condition

You want characteristic TANDEM\_SADDLE hidden if characteristic MODEL does not have a value. Add the following condition to the source code:

```
MODEL eq 'Tandem' and Specified MODEL
```

There are now 2 conditions to fulfill for this precondition:

1. Characteristic MODEL must have a value.
2. Characteristic MODEL must have the value 'Tandem'.

## Selection Conditions

### Use

You can use selection conditions to ensure that all the objects relevant to a variant are selected:

- Selection conditions determine which variants require a specific component or operation
- Selection conditions determine when it is mandatory to assign a value to a characteristic

You can allocate selection conditions to the following objects:

- Characteristics
- BOM items
- Operations in task lists
- Sub-operations
- Sequences of operations
- Production resources/tools (PRTs)

### Features

A selection condition is fulfilled if the condition in it is unambiguously true.



A selection condition is fulfilled if the value in the condition is set for the characteristic.

A selection condition is not fulfilled if:

- a) A different value is set for the characteristic
- b) No value is set for the characteristic

## Selection Conditions for a BOM Item and Operation

## Selection Conditions for a BOM Item and Operation

Characteristic HANDLEBAR is assigned to configurable material BIKE. Characteristic HANDLEBAR has the following values:

HANDLEBAR	Racing
	MOUNTAIN_HANDLEBAR
	STANDARD_HANDLEBAR

Each handlebar has its own component in the BOM and its own operation in the routing. A selection condition is allocated to each component and each operation. This selection condition determines which components and operations are selected for a variant.

## BOM of BIKE

Item	Component	Selection condition
0010	Racing	Handlebar = 'Racing'
0020	MOUNTAIN_HANDLEBAR	Handlebar = 'Mountain'
0030	STANDARD_HANDLEBAR	Handlebar = 'Standard'

## Routing of BIKE

Operation	Description	Selection condition
0010	Mount Racing handlebar	Handlebar = 'Racing'
0020	Mount Mountain handlebar	Handlebar = 'Mountain'
0030	Mount Standard handlebar	Handlebar = 'Standard'

## Procedure

1. You create 3 selection conditions.
2. Enter the appropriate source code for each selection condition:
  - `Handlebar eq 'Racing'`
  - `Handlebar eq 'Mountain'`
  - `Handlebar eq 'Standard'`

### Selection Conditions for a BOM Item and Operation

3. Allocate each selection condition to the appropriate BOM component and the appropriate operation in the routing.

When you configure the bike, the correct component and operation for each value of characteristic HANDLEBAR are selected automatically.

**Selection Condition for a Characteristic**

## Selection Condition for a Characteristic

Characteristic MODEL is assigned to configurable material BIKE.

Then characteristic TANDEM\_SADDLE is assigned to configurable material BIKE. This characteristic has a precondition (refer to [Example: Precondition for a Characteristic \[Page 76\]](#)).

Characteristic	Values	Condition
MODEL	Racing	
	Standard	
	Mountain	
	Tandem	
TANDEM_SADDLE		Precondition: MODEL = TANDEM
		Selection condition: MODEL = TANDEM

You then want to assign a selection condition to the characteristic, which determines that you must assign a value to the characteristic if MODEL 'Tandem' is selected.

## Procedure

1. You create a selection condition.
2. This selection condition has the following source code:

```
MODEL eq 'Tandem'
```

3. Allocate the selection condition to characteristic TANDEM\_SADDLE.

## Result

If you enter value 'Tandem' for characteristic MODEL when you configure material BIKE, you fulfill 2 conditions:

- The precondition displays the characteristic.
- The selection condition makes you assign a value to the characteristic.

If you do not assign a value to characteristic TANDEM\_SADDLE, you automatically see the characteristic when you exit the configuration editor, so that you can assign a value.

You can only leave the configuration editor if you do one of the following:


- Assign a value to the characteristic
- Change the status of the configuration



## Procedures

### Definition

You can use procedures to infer values for characteristics – they are like actions in this respect. However, there are some important differences:

Procedures	Actions
	 Actions are an old dependency type. You can almost always use procedures to do the same thing.
Procedures can overwrite default values that are set by other procedures.	Actions cannot overwrite values that are set by other actions.
Procedures can set default values for a characteristic, which can be overwritten by the user.	The user cannot overwrite values that are set by an action.
If several procedures are assigned to an object, you can define a processing sequence (see <a href="#">Processing Sequence of Procedures [Page 83]</a> ).	You cannot influence the sequence in which actions are processed.

You can assign procedures to the following objects:

- The characteristic value that triggers the procedure
- The characteristic that triggers the procedure
- The configuration profile of the configurable object



It is easier to assign procedures to the configuration profile, so that they are all in one place.

- BOM items – to change the component quantity, for example
- Operations in task lists – to change the standard values, for example

### Use

- If you use a procedure to infer a value for a characteristic, you enter the variable **\$SELF** before the characteristic.
- Procedures can overwrite values that are set by other procedures.
- Procedures are always used for pricing (see [Variant Conditions \[Page 211\]](#)).

Additional keywords for procedures:

- Set default values:

**Procedures**

`$SET_DEFAULT ($SELF, <characteristic>, <term>)`

- Delete default values:

`$DEL_DEFAULT ($SELF, <characteristic>, <term>)`

- Produce a sum of the values for a characteristic in a multi-level configuration:

`$SUM_PARTS ($SELF, <characteristic>)`

- Add the components of a BOM together:

`$COUNT_PARTS (<$SELF>)`

- Surcharge for variant conditions:

`$SET_PRICING_FACTOR ($SELF, <characteristic>, <variant key>, <factor>)`

In procedures, you can use the following keywords, which are not allowed in actions:

- NOT SPECIFIED
- NOT TYPE\_OF
- <multiple-value characteristic>NE<value>

**See also:**

[Changing Master Data with Dependencies \[Page 96\]](#)

## Processing Sequence of Procedures

### Use

If an object has more than one procedure, you can define a processing sequence.

When you allocate a procedure to an object, you see a field for defining the point in the sequence when the procedure is processed.

The sort sequence only applies to the procedures for that object.



A configurable material has characteristics COLOR and PRESSURE. The following procedures are allocated to the configuration profile of the material.

0010 \$SELF.COLOR = 'GREEN' IF PRESSURE >= 10

0020 \$SELF.COLOR = 'YELLOW' IF PRESSURE >= 50

0030 \$SELF.COLOR = 'RED' IF PRESSURE >= 100

The sort sequence ensures that pressure greater than or equal to 100 always sets the color 'red'.

If procedures are assigned to characteristics and values, they are processed in the sequence of the characteristics and by sort field within a characteristic.

Before the procedures are processed, all procedure inferences for the current object are deleted.

#### See also:

[Processing Sequence of Dependencies \[Page 242\]](#)

## Inferring a Characteristic Value with Procedures (Example)

## Inferring a Characteristic Value with Procedures (Example)

### Use

Configurable material BIKE has the following characteristics:

Characteristic	Values	Weight calculation
WEIGHT (3 figures, 1 decimal place)	-	
FRAME	Aluminum Steel	22.05 lb 30.86 lb
EXTRAS (multiple-value)	Mudguard Luggage rack	0.5 kg 1.0 kg

### Procedure

1. Create a procedure for the weight of the BIKE, depending on the frame.
2. This procedure has the following source code:
 

```
$SELF. WEIGHT = 10 if FRAME = 'Aluminum' ,
$SELF. WEIGHT = 14 if FRAME = 'Steel' .
```
3. Allocate the procedure to the configuration profile of material BIKE.

The weight of a bicycle increases if you select additional extras, such as mudguard or luggage rack.

1. You define a procedure with the following source code:
 

```
$SELF.WEIGHT = $SELF.WEIGHT + 0.5 if EXTRAS = 'Mudguard' ,
$SELF.WEIGHT = $SELF.WEIGHT + 1 if EXTRAS = 'Luggage rack'
```
2. Allocate the procedure to the configuration profile of material BIKE.

### Result

1. When you configure the bike, the value 'Aluminum' is selected for characteristic FRAME. This triggers the first procedure, which sets the value 10 kg as the WEIGHT.
2. Characteristic EXTRAS has values 'Mudguard' and 'Luggage rack'. This triggers the second procedure, which increases the value of characteristic WEIGHT to 11.5 kg.



You cannot use an action to change the value already set for characteristic WEIGHT.

## Built-In Function \$COUNT\_PARTS

### Use

You use built-in function \$COUNT\_PARTS in a procedure, to produce a sum of the values of a characteristic and set this sum as a value for a characteristic.

The procedure can only be processed properly if all the relevant BOM components are known in the configuration. For this reason, only process the procedure at the end of the BOM explosion.



```
$SELF.<characteristic> = $COUNT_PARTS (<object variable>)
```

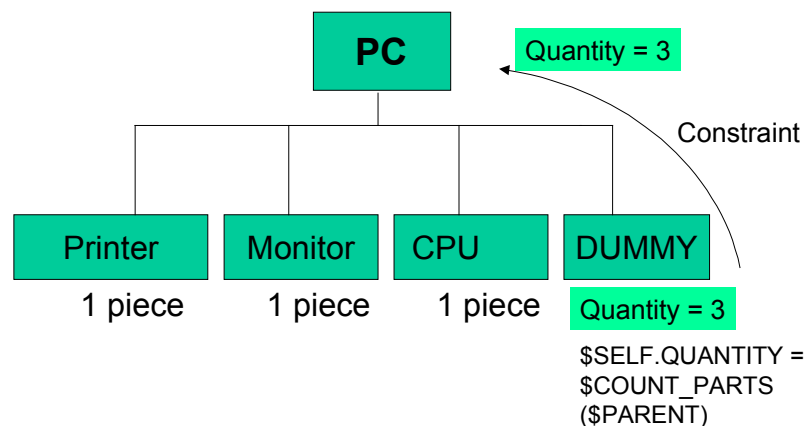


You can create a configurable dummy material to use as the last item in a BOM. Assign characteristic QUANTITY to the material. Allocate a procedure with the following source code to the BOM item:

```
$SELF. QUANTITY = $COUNT_PARTS ($PARENT)
```

\$SELF is characteristic QUANTITY of the BOM item that has the procedure. \$COUNT\_PARTS adds together the quantities of all components of the BOM of superior material \$PARENT, to infer a value for this characteristic.

You use a constraint to transfer this value to the superior characteristic.



---

**Built-In Function \$COUNT\_PARTS****Prerequisites**

You can only use this function of a procedure for an object that can be configured interactively on multiple levels.

The summation assumes that all components are counted in the same unit of measure (for example, piece).

**Features**

- The procedure only reads components on the first level of the BOM.
- You cannot exclude any BOM components from the sum total.

## Built-In Function \$SUM\_PARTS

### Use

You use built-in function \$SUM\_PARTS in a procedure, to produce a sum of the values of a characteristic, across several configurable materials in a configuration structure.

The procedure can only be processed properly if all the relevant BOM components are known in the configuration. For this reason, only process the procedure at the end of the BOM explosion.



```
$SELF.<characteristic> = $SUM_PARTS (<object variable>,  
<numeric characteristic>)
```



To add together the weights of the different components in a BOM, create characteristic WEIGHT and allocate it to all the configurable materials that you want to include in your calculation.

Create a configurable material as a placeholder and enter this material as the last item in the BOM. Assign characteristic WEIGHT is to the material. Allocate a procedure with the following source code to the BOM item:

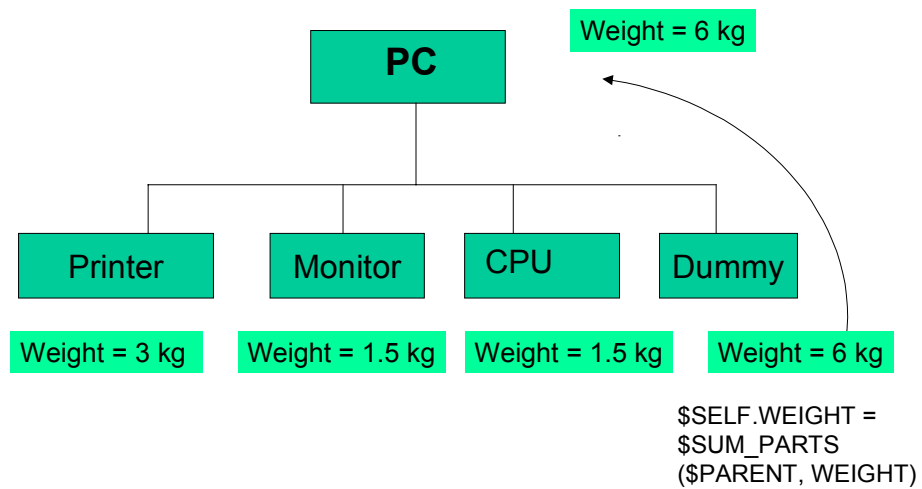
```
$SELF.WEIGHT = $SUM_PARTS ($PARENT, WEIGHT)
```

\$SELF is characteristic WEIGHT of the BOM item that has the procedure.

\$SUM\_PARTS adds together the weights of all components of the superior material \$PARENT, to infer a value for the weight of the placeholder material.

You use a constraint to transfer this value to the superior material.

## Built-In Function \$SUM\_PARTS



## Prerequisites

You can only use this function of a procedure for an object that can be configured interactively on multiple levels.



## Setting Default Values with Procedures

### Use

You can use procedures to set default values for a characteristic, which can be overwritten by the user.

Use the following language element to set default values:

```
$SET_DEFAULT ($SELF, <characteristic>, <string>)
```

As the string, you can enter either a characteristic value or, for numeric characteristics, a calculation.



Setting a default value for a numeric characteristic:

1. A configurable material has characteristics HEIGHT and WIDTH.
2. The height is usually one and a half times the width, so you want to set this as a default value for characteristic HEIGHT.
3. You define a procedure with the following source code:

```
$SET_DEFAULT ($SELF, HEIGHT, 1.5 * $SELF.WIDTH)
```
4. Allocate the procedure to characteristic WIDTH or the configuration profile of the material.
5. As soon as you enter a value for WIDTH, the procedure is triggered and sets the value of HEIGHT as one and a half times the WIDTH. You can change this value manually.

### Features

- If the value that triggers the procedure is deleted, this does not delete the default value.
- If a default value conflicts with a 'hard' value that is already set, (for example, a user entry), the default value is ignored.
- The user can overwrite default values, but cannot delete them.

If you delete the default value, it reappears as soon as you press enter, because the dependencies are processed again. You cannot define that no default value is entered for the characteristic.

You have the following options for deleting a default value:

- For single-value characteristics, the statement that no value is set can be expressed using an explicit additional value. The user can use this value to overwrite the default value.
- For multiple-value and restrictable characteristics, you can create an additional characteristic, 'USE\_DEFAULTS', and use procedures with the expressions `$SET_DEFAULT` and `$DEL_DEFAULT`, to set or delete default values, according to the value assigned to this characteristic.

**See also:**

---

**Setting Default Values with Procedures**

[Deleting Default Values with Procedures \[Page 91\]](#)

## Deleting Default Values with Procedures

### Use

You cannot manually delete default values that were set by a procedure. However, you can use another procedure to delete a default value.

Use the following language element to delete default values:

```
$DEL_DEFAULT ($SELF, <characteristic>, <string>)
```



Deleting a default value for a numeric characteristic:

Characteristic HEIGHT has a default value that was set by a procedure (refer to [Setting Default Values with Procedures \[Page 89\]](#)).

1. If characteristic DEL\_DEFAULTVALUE has the value 01, this default value is deleted.
2. You define a procedure with the following source code:  

```
$DEL_DEFAULT ($SELF, HEIGHT, 1.5 * $SELF.WIDTH)
```
3. You allocate the procedure to characteristic value 01 of characteristic DEL\_DEFAULTVALUE.
4. If you select value 01 for characteristic DEL\_DEFAULTVALUE, the default value for characteristic HEIGHT is deleted.

### Prerequisites

The default value you want to delete was set by a procedure.

If the value set by the procedure is not a default value, \$DEL\_DEFAULT is ignored.

### Features

When the default value is deleted, this does not automatically delete other values that were inferred from the default value.

**Assigning Object Dependencies**

## Assigning Object Dependencies

You can assign object dependencies to the following objects:

Configuration profile	<ul style="list-style-type: none"><li>– Dependency nets</li><li>– Procedures</li></ul>
Characteristics	<ul style="list-style-type: none"><li>– Procedures</li><li>– Preconditions</li><li>– Selection conditions</li></ul>
Characteristic values	<ul style="list-style-type: none"><li>– Procedures</li><li>– Preconditions</li></ul>
BOM items	<ul style="list-style-type: none"><li>– Procedures</li><li>– Selection conditions</li></ul>
Operations in task lists	<ul style="list-style-type: none"><li>– Procedures</li><li>– Selection conditions</li></ul>
Production resources/tools (PRTs)	<ul style="list-style-type: none"><li>– Procedures</li><li>– Selection conditions</li></ul>

### Procedures

If you want to use procedures to influence the characteristic values on the value assignment screen, you assign them to the configuration profile, characteristic, or characteristic value.



The result is the same, whether procedures are assigned to the configuration profile, the characteristic, or the characteristic value. However, it is easier to administer procedures if you assign them all to the configuration profile.

## Assigning Object Dependencies

### 1. Configuration profile

Material	BIKE
Profile name	Profile_1
Dependency	\$SELF.Saddle = 'Leather' if Model = 'Racing'

### 2. Characteristic

Model of Bike	\$SELF.Saddle = 'Leather' if Model = 'Racing'
<input type="radio"/> Racing <input type="radio"/> Mountain <input type="radio"/> Tandem	

### 3. Triggering value

Model of Bike	\$Self.Saddle = 'Leather'
<input type="radio"/> Racing <input type="radio"/> Mountain <input type="radio"/> Tandem	

If you want to use procedures to change fields in BOM items, operations in task lists, or PRTs, you assign them to the relevant items, operations, and PRTs.

### Procedure for BOM Item

Material	Bike		
Item	Component	Quantity	Unit
0010	Frame	1	EA
0020	Handlebar	1	EA
0030	Saddle	1	EA
0040	Brake	2	EA
		\$Self.quantity = 3 if Backpedal = 'YES'	

## Assigning Object Dependencies

Procedures work locally (object &SELF) for:

- Configurable materials
- BOM items
- Operations in task lists
- PRTs

You cannot use procedures to infer values for the characteristics of other configurable materials. To do this, you use dependency nets.

## Preconditions and Selection Conditions

You assign preconditions and selection conditions to the characteristic or characteristic value to which the condition applies.

The diagram shows a 'Value Assignment' form with three characteristics: Model, Tandem\_Saddle, and Gears. Each characteristic has a dropdown menu. The 'Model' dropdown is open, showing options: Racing, Mountain, and Tandem. The 'Tandem\_Saddle' dropdown is open, showing the option: Model = 'Tandem'. The 'Gears' dropdown is open, showing options: 12 gears, 18 gears, and 21 gears. The 'Gears' dropdown also shows the condition: Model = 'Racing'.

Characteristic	Value	Condition
Model	Racing	
Model	Mountain	
Model	Tandem	
Tandem_Saddle	Model = 'Tandem'	
Gears	12 gears	
Gears	18 gears	
Gears	21 gears	Model = 'Racing'



Assignment to affected characteristic  
Assignment to affected characteristic value

You assign selection conditions to the BOM items, operations, and PRTs to which the condition applies.

Assigning Object Dependencies

Material	Bike	
Operation	Description	
0010	Install racing handlebar	Handlebar = 'Racing_handlebar'
0020	Install mountain bike hbar	Handlebar = 'Mountain_handlebar'
0030	Install standard handlebar	Handlebar = 'Standard_handlebar'

## Changing Master Data with Dependencies

## Changing Master Data with Dependencies

### Purpose

In variant configuration, you can use reference characteristics with dependencies to change field values in bills of material (BOMs), task lists, and master recipes.



Do not use object dependencies to change control fields, such as internal counters, node numbers, and item categories of BOM items.

For example, you can change the following fields in BOMs, task lists, and master recipes:

STPO (BOM)	Item text (POTX1 and POTX2)
	Variable-size item sizes 1–3 (ROMS1–3)
	Number of variable-size items (ROANZ)
	Variable-size item quantity (ROMEN)
	Component quantity (MENGE)

PLPOD (task list and master recipe)	Activity types (LAR01–06)
	Standard values (VGW01–06)
	Unit of measure (VGE01–06)
	Work center (ARBPL)
	Operation description (LTXA1, LTXA2)

PLFLD (sequence of operations)	Lot size from (LOSVN)
	Lot size to (LOSBS)

PLFHD (production resources/tools)	Quantity (MGVGW)
	Quantity formula (MGFORM)
	Required quantity (EWVGW)
	Required quantity formula (EWFORM)

### Process Flow

1. In characteristics maintenance, create a characteristic with a reference to the table field you require.



## Changing Master Data with Dependencies

2. Assign the reference characteristic to the variant class. In characteristics maintenance, you can define the reference characteristic as hidden.
3. Create an action or procedure that uses the reference characteristic to refer to the table.



We recommend that you **always** use **procedures** to change master data, not actions.

4. Allocate the object dependency to a BOM item, an operation, an operation sequence, or a production resource/tool (PRT).

## Result

When the action or procedure is triggered, it changes the field value referred to in the reference characteristic.

### See also:

[Creating a Reference Characteristic \[Ext.\]](#)

[Reference Characteristics \[Ext.\]](#)

PP Routings

[Creating Global Dependencies for an Operation \[Ext.\]](#)

[Creating Local Dependencies for an Operation \[Ext.\]](#)

PPPI Master recipes

[Maintaining Global Object Dependencies \[Ext.\]](#)

[Maintaining Local Object Dependencies \[Ext.\]](#)

## Reference Characteristics in Dependencies

## Reference Characteristics in Dependencies

### Use

You can use reference characteristics in actions and procedures to change the values in master data fields.



We recommend that you **always** use **procedures** to change master data, not actions.

In the conditional part of actions and procedures, and in preconditions and selection conditions, you can have read-only access to a table.

### MDATA

If you are using a reference characteristic in an action or procedure to change the value of a master data field, enter MDATA before the characteristic, because reference characteristics have no initial value in configuration.

If you try to change the master data field without using the keyword MDATA, the system does not recognize the characteristic as having a value. Processing is terminated.



You have created reference characteristic QUANT, which refers to the component quantity in a BOM item.

You use this reference characteristic in an action or procedure. You want to increase the component quantity by 1:

```
$SELF.QUANT = MDATA $SELF.QUANT + 1
```

The expression MDATA refers to the quantity in the BOM. This quantity is increased by 1.

The reference characteristic must be referred to with the variable \$SELF, because the reference characteristic refers to the BOM item currently being processed.

Reference Characteristics in Procedures	Reference Characteristics in Actions
MDATA lets you access the original value of a master data field and change it. You can change the value again. Since you then access the changed value, the expression MDATA is no longer required.	MDATA lets you access the original value of a master data field and change it. You cannot change the value again.

### Restrictions

MDATA cannot be used with a characteristic that refers to structure SDCOM of variant conditions.

### Reference Characteristics in Dependencies

Characteristics that refer to this table are multiple-value characteristics, so the expression MDATA cannot express a single value. Besides, this table contains transaction data rather than master data.

## Master Data References in Bills of Material

## Master Data References in Bills of Material

Configurable material BIKE has the following characteristic:

Characteristic	Values
BACKPEDAL	Yes
	No

If a bike is ordered with a backpedal brake, the number of brakes increases to 3.

BOM of BIKE

Item	Component	Qty	Action
0030	BRAKE	2	Quantity = 3 if Backpedal

## Procedure

1. Create a reference characteristic called QUANTITY with the following master data reference:  
Table STPO field MENGE
2. Create a procedure with the following source code:  

```
$SELF.QUANTITY = 3 if BACKPEDAL = 'Yes'
```
3. Allocate the procedure to BOM item 0030 (component BRAKE).

## Result

When you configure the bike, if you set value 'Yes' for characteristic BACKPEDAL, the quantity for component BRAKE increases to 3.

## Master Data References in Task Lists

Configurable material BIKE has the following characteristic:

Characteristic	Values
COLOR	MG (Mirror green)
	KR (Kansas red)
	BG (Baltimore gray)
	FL (French lavender)

You need a special painting procedure for the color 'French lavender'. This increases the setup time of operation 'Paint bike' from 30 minutes to 60 minutes.

Routing of BIKE

Operation	Description	Procedure
0040	Paint bike	Setup time + 30 mins for color 'French lavender'

### 1. Create Procedure

1. Create reference characteristic SETUP\_TIME with the following master data reference:  
Table PLPOD field VGW01
2. Create a procedure with the following source code:  
`$SELF.SETUP_TIME = MDATA $SELF.SETUP_TIME + 30 IF COLOR = 'FL'`
3. Assign the procedure to operation 'Paint bike' in the routing for the bike.

### 2. Create Procedure

For the color 'French lavender', create characteristic METALLICEFFECT:

Characteristic	Values
METALLICEFFECT	Yes
	No

If you want metallic paintwork, the setup time increases by another 30 minutes.

Routing of BIKE

Operation	Description	Procedure
0040	Paint bike	Setup time + 30 mins for color 'French lavender' Setup time + 30 mins for metallic effect

## Master Data References in Task Lists

1. Create a procedure with the following source code:

```
$SELF.SETUP_TIME = $SELF.SETUP_TIME + 30 IF METALLICEFFECT = 'Yes'
```



You do not use the keyword MDATA in this procedure, because you are accessing the changed value.

2. Assign the procedure to operation 'Paint bike'.

## Result

To check the result in the configuration simulation:

1. Assign value 'FL' to characteristic COLOR.
2. Choose *Result*.
3. On the result screen, select header material BIKE and choose *View* → *Objects* → *Task list*.

You see the routing for material BIKE.

4. Select operation 'Paint bike' and choose *Information*.

You see that the standard value has increased from 30 minutes to 60 minutes.

5. Choose *Characteristics* and assign value 'Yes' to characteristic METALLICEFFECT.
6. Display the result.

The standard value has increased to 90 minutes.

### See also:

[Reference Characteristics in Dependencies \[Page 98\]](#)

## Changing the Weight in the Sales Order

The weight of header material PC is calculated during configuration, then automatically copied to the sales order.

Configurable material PC has characteristic WEIGHT.

The weight of the PC is calculated by function \$SUM\_PARTS (see [Built-In Function \\$SUM\\_PARTS \[Page 87\]](#)).

This value is then copied to the sales order for the PC as the gross weight.

### Procedure

1. Create reference characteristic GROSS\_WEIGHT with the following table reference:  
Structure VCSD\_UPDATE field BRGEW.  
Set the *No display* indicator for the reference characteristic, so that it is not displayed in the configuration editor.
2. Assign the reference characteristic to the variant class for the PC.
3. Create a procedure with the following source code:  

```
$SELF.GROSS_WEIGHT = $SELF.WEIGHT
```
4. Allocate the procedure to the configuration profile of the PC.

### Result

When you configure the PC, the value of characteristic WEIGHT is copied to the *Gross weight* field in the SD document.

**Example: Shape and Size Variants****Example: Shape and Size Variants**

You can use table references in actions and procedures to calculate the variable-size item quantity for variable-size items.

**Requirements**

Configurable material DOOR has the following characteristics:

Characteristic	Value	Dependent values
DOOR_WIDTH	0,6100	X
	0,7350	
DOOR_HEIGHT	1,980	X

BOM:

Item	Component	Item category	Dependencies
0030	DOOR_PANEL_01	L (stock item)	Selection condition: DOOR_WIDTH = 0.6100 and DOOR_HEIGHT = 1.980
0040	DOOR_PANEL_02	L (stock item)	Selection condition: DOOR_WIDTH = 0,7350 and DOOR_HEIGHT = 1.980
0050	DOOR_PANEL_S	R (raw material)	Selection condition: DOOR_WIDTH not 0.6100 or 0.7350 DOOR_HEIGHT not 1.980  Procedure to infer sizes and calculate variable-size item quantity

If you enter the standard sizes for DOOR\_HEIGHT and DOOR\_WIDTH, either DOOR\_PANEL\_1 or DOOR\_PANEL\_2 is selected from the BOM. However, if you enter any other size, the component DOOR\_PANEL\_S (special) is selected.

The sizes for the special door panel are inferred using an procedure.

**Procedure****Create Selection Conditions**

1. Create the following selection conditions for the individual door panels:

DOOR\_PANEL\_01:

`DOOR_WIDTH = 0.6100 and DOOR_HEIGHT = 1.980`

DOOR\_PANEL\_02:

`DOOR_WIDTH = 0.7350 and DOOR_HEIGHT = 1.980`



Example: Shape and Size Variants

DOOR\_PANEL\_S:

```
NOT ((DOOR_WIDTH = 0.6100 and DOOR_HEIGHT = 1.980)
or (DOOR_WIDTH = 0.7350 and DOOR_HEIGHT = 1.980))
```

2. Allocate the selection conditions to the individual BOM items.

When you configure the door, the appropriate door panel is selected for the DOOR\_WIDTH and DOOR\_HEIGHT you enter.

## Create Procedure

1. Create reference characteristics that refer to the fields *Size 1*, *Size 2*, and *Variable-size item quantity* in the BOM item:

Characteristic	Table	Field
SIZE_1	STPO	ROMS1
Size_2	STPO	ROMS2
VSI_qty	STPO	ROMEN

2. Create a procedure with the following source code:

```
$SELF.SIZE_1 = DOOR_WIDTH;
$SELF.SIZE_2 = DOOR_HEIGHT;
$SELF.VSI_QTY = DOOR_WIDTH * DOOR_HEIGHT.
```

3. Allocate the procedure to BOM item DOOR\_PANEL\_S.

## Result

When you configure a door, the values you assign to characteristics DOOR\_WIDTH and DOOR\_HEIGHT trigger the procedure, which sets values for *Size 1* and *Size 2* and calculates the variable-size item quantity.

## Constraints

# Constraints

This dependency type is mainly for intensively interactive configuration tasks and for configuration tasks in which you need to take into account the dependencies between the characteristics of several objects. The main purpose of a constraint is to monitor the consistency of a configuration.

Constraints have the following distinguishing features:

- You can use constraints to describe the dependencies between completely different objects and their characteristics.
- Constraints are used to store information on which conditions must be fulfilled if the configuration is to be consistent.
- Constraints are not directly allocated to individual objects. They are grouped together to form dependency nets and allocated to a configurable material in the configuration profile.
- In constraints, you enter objects in their general form of expression, without using \$SELF, \$ROOT, or \$PARENT to identify objects. As a rule, you refer to objects in constraints by entering the class to which the objects are allocated.
- Constraints are declarative dependencies. The processing sequence of constraints and the point in time when constraints are processed are not relevant.
- Constraints are not processed in a specific order. You cannot determine when a specific constraint is used.

In any processing situation, a constraint is only processed once. If a value that is relevant to the constraint is changed, the constraint is triggered again.

## Structure of Constraints

There are four sections in a constraint. Each part is identified by a keyword. The keyword is followed by a colon. Each section ends with a period.

- **OBJECTS:**  
In this section, you enter the objects that are relevant to the constraint. You **must** enter the relevant objects in all constraints. You can also define variables for objects or characteristics.
- **CONDITION:**  
The condition entered here must be fulfilled in order for the constraint to be used. You do not need to enter a condition in a constraint. You can leave out the keyword **CONDITION:** if required. However, if you enter the keyword you must enter a condition.
- **RESTRICTIONS:**  
In this section, you enter the relation that must exist between the objects and characteristics if the configuration is to be consistent. You **must** enter a restriction in a constraint.
- **INFERENCES:**

## Constraints

In this section, you enter the characteristics for which characteristic values are to be inferred. The main purpose of constraints is to check the consistency of a configuration. Usually, values are only inferred if you make an entry in this section.

For reasons of performance, only use constraints to infer values if it is really necessary.

Constraints are grouped together to form dependency nets. The dependency net is allocated to a configurable material in the configuration profile.

For more information on constraints, refer to:

- [Constraints: Referring to Objects \[Page 108\]](#)
- [Constraints: Entering Conditions \[Page 111\]](#)
- [Constraints: Restrictions \[Page 114\]](#)
- [Constraints: Inferences \[Page 116\]](#)
- [Constraints: Restricting the Value Set \[Page 118\]](#)
- [Example: Networks in a Company \[Page 124\]](#)
- [Relationship between Operating System and Server \[Page 128\]](#)
- [Relationship between Operating System of Server and Operating System of Workstation \[Page 129\]](#)
- [Relationship between Company Network Server and Departmental Network Server \[Page 131\]](#)
- [Relationship between LAN Type and Server Processor \[Page 133\]](#)
- [Relationship between Cable Type and LAN Type \[Page 134\]](#)
- For information on how to create dependency nets and constraints, see [Creating a Dependency Net \[Page 120\]](#).

## Constraints: Referring to Objects

## Constraints: Referring to Objects

You must declare all objects that are relevant to a constraint in the OBJECTS: section of the constraint.



**All** of the characteristics in the OBJECTS section of the constraint must be used in the other sections of the constraint (RESTRICTIONS and CONDITION). If you declare a characteristic that is not used in the constraint, the constraint is not processed.

### Class and Class Type

The usual way to identify an object in a constraint is by entering the class and class type. If you want to use a constraint to refer to a PC classified in class PC of class type 300, you enter the PC as follows:

```
(300) PC
```

### Object Key

If you want to refer to a specific material, you enter the identification for the object type, the class type, and the object key.

- MATERIAL in this example is the object identification for materials. You define the object identification for referring to objects in dependencies in Customizing for *Classification*, by choosing *Maintain object types*.
- The object key for materials is the material number. The object key is also defined in Customizing for *Classification*. In the step *Maintain object keys*, you define which fields are used to refer to an object. You also enter the identification that is used to refer to the fields. The material number is referred to by formal parameter NR. An object key can consist of several partial keys, as is the case for documents:

```
(Document) (017) (TYPE = 'DRW', VERSION = '00', PART = '000', NR = 'D4545')
```

- For the class type, enter the class type in which the object is classified. If the object is not classified, do not enter a class type. However, you must enter empty parentheses as a place holder.

```
(Material) () (NR = 'MAT_A')
```

### Declaring Several Objects

If you want to refer to other objects in your constraint, you must declare these objects in the constraint, too. You use a comma to separate the individual objects.

OBJECTS:

```
PC IS_A (300) PC,
```

```
PT IS_A (001) PRINTER,
```

```
EX IS_A (001) EXTRAS
```

## Constraints: Referring to Objects

In this constraint, the objects referred to are PC, Printer, and Extras. The objects are referred to by their class and class type.

### Defining Variables

- If you use a class to refer to an object, you use the expression **IS\_A** to define a variable:

```
PC IS_A (300) PC
```

- If you refer to an object using the object key, and do not enter a class, you use the expression **IS\_OBJECT** to define a variable:

```
O IS_OBJECT (Material) (001) (NR='M4711')
```

- If you use the object key to refer to an object, you must define a variable. If you use a class to refer to an object, defining a variable is optional.
- You can enter a ? to make your variables easier to identify:

```
?PC IS_A (300) PC
```

- The variables referred to in a constraint must already be known when you create a constraint.



Usually, it is hard to use a constraint to calculate the total weight of a component by calculating the sum of the weights of its individual parts, because all the relevant objects must be entered in the constraint. If one of the objects is not selected in the configuration process, the constraint cannot be used because one of the input parameters is missing. This means that you can only define a constraint for this task if all the objects concerned must always be selected.

### Referring to Characteristics

Characteristics are always referred to in the context of the object to which they belong. Characteristics are identified by their language-independent name. If no variables are defined for objects in the OBJECTS section of the constraint, you must enter a complete expression consisting of class and class type before the characteristic:

```
(300) PC.HARD_DISK = '1275'
```

This shows that the characteristic belongs to an object in class **PC** of class type **300**.

If you have defined a variable for an object, you see the variable instead of the object:

```
PC.HARD_DISK = '1275'
```



Characteristic values are only case sensitive if the characteristic format defines them as such.

### Defining Variables

In the OBJECTS section, you can also define variables for characteristics. These variables are connected to the object to which they belong using the keyword **WHERE**.

```
PC IS_A (300) PC
WHERE HD = HARD_DISK
```

### Constraints: Referring to Objects

This shows that the characteristic belongs to class **PC**, so the characteristic is referred to afterwards with just the variable HD:

```
HD = '1275'
```

You can also define variables for several characteristics. Use a semicolon to separate the characteristics from each other:

```
PC IS_A (300) PC  
WHERE HD = HARD_DISK; C = CASING; CPU = CPU
```

### Multiple-Value Characteristics

If you want your constraint to refer to several values of a multiple-value characteristic, you must define several variables:

```
OBJECTS:  
(300) PC  
Where Var1 = Color_multi; Var2 = Color_multi  
  
CONDITION:  
Var1 = 'red' and Var2 = 'yellow'  
  
RESTRICTIONS:  
false
```

This constraint refers to 2 values of multiple-value characteristic COLOR\_MULTI. This is why it is necessary to declare two variables, one for each value.

If values 'red' and 'yellow' are set for characteristic COLOR\_MULTI, the constraint triggers an inconsistency message.

## Constraints: Entering Conditions

In the condition section of a constraint, identified by the keyword **CONDITION**, you define when a constraint is valid. This condition acts as a filter. Under **RESTRICTIONS**, you enter the consistency checks that a constraint is to make.

For example, you can define that a constraint is only processed if value '586' is assigned to characteristic CPU for material 'PC':

**OBJECTS:**  
PC IS\_A (300) PC

**CONDITION:**  
PC.CPU = '586'



If you enter more than one condition, there is an **AND** relationship between these conditions:

**CONDITION:**  
PC.CPU = '586' and PC.HARD\_DISK = '1620'

## Conditions in the Restrictions Section

Some simple conditions can also be entered directly in the restrictions section:

**OBJECTS:**  
PC IS\_A (300) PC

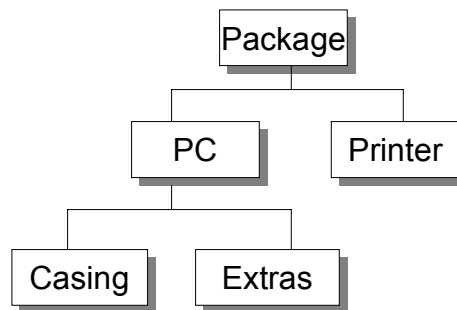
**RESTRICTIONS:**  
PC.HARD\_DISK = '1620' if PC.CASING = 'Tower',  
PC.HARD\_DISK = '850' if PC.CASING = 'Minitower'

The hard disk can only have the value '1620' for a tower, or '850' for a minitower.

## PART\_OF and SUBPART\_OF

If a constraint refers to several objects that are part of a BOM, you can use the **CONDITION** section to define that the constraint is only valid if the objects belong to the BOM of the superior material.

## Constraints: Entering Conditions



PART\_OF (CASING, PC) is true.  
 PART\_OF (CASING, PACKAGE) is false.  
 SUBPART\_OF (CASING, PC) is true.  
 SUBPART\_OF (CASING, PACKAGE) is true.

The example shows that a constraint is only processed if the casing is a component of the BOM for the PC. This condition is entered under **CONDITION** using the expression **PART\_OF**:

**OBJECTS:**

PC IS\_A (300) PC,  
 C IS\_A (300) CASING

**CONDITION:**

**PART\_OF (C, PC)**

The constraint is only valid if the casing is part of the PC.

The condition in the constraint is very important, because without the condition the constraint would be valid for all combinations including a PC and a casing, even if the casing is not in the BOM for the PC.

If you want a constraint to refer to the components of sub-assemblies, you use the expression **SUBPART\_OF**.

**OBJECTS:**

P IS\_A (300) PACKAGE,  
 C IS\_A (300) CASING

**CONDITION:**

**SUBPART\_OF (C, P) .**

In this example, the condition **PART\_OF (C, P)** is false, because the casing is not directly a component of the package.



## Referring to BOM Items

You can also refer to a specific BOM item in the condition section. For example, if you want the casing to be in a specific BOM item, you enter the BOM item.

### OBJECTS:

PC IS\_A (300) PC,

C IS\_A (300) CASING

### CONDITION:

PART\_OF (C, PC, '0050')

The constraint checks whether the casing is item 50 of the BOM for the PC.

**Constraints: Restrictions**

## Constraints: Restrictions

In the RESTRICTIONS section of a constraint, you enter the consistency checks that the constraint is to make. The constraint returns an inconsistency if the restrictions section is not true.



You cannot enter non-declarative expressions, such as NOT SPECIFIED (refer to [Declarative Object Dependencies \[Page 140\]](#)).

Example:

You can use a constraint to define that the print drive for a WIN95 printer must be selected if operating system WIN95 is selected for a PC:

```
OBJECTS:
(300) PC
where OS = OPERATING_SYSTEM,
(300) PRINTER
where PD = PRINT_DRIVE

RESTRICTIONS:
PD = 'Drive_WIN95' if S eq 'WIN95'
```

You can also enter more than one restriction. You enter a comma to separate the different restrictions. There is an AND relationship between the restrictions. OR relationships are not supported.

Example:

```
RESTRICTIONS:
PD = 'Drive_WIN95' if OS eq 'WIN95',
PD = 'Drive_OS/2' if OS eq 'OS/2',
PD = 'Drive_NT' if OS eq 'NT'.
```

## False

You can also use the statement FALSE as a restriction. This means that situations described in the condition section are inconsistent. If these situations occur, an inconsistency message is returned.

Example:

```
OBJECTS:
PC IS_A (300) PC

CONDITION:
PC.CPU = '486' and PC.EXTRAS = 'CO_processor'.

RESTRICTIONS:
FALSE.
```

This constraint returns an inconsistency as soon as the value '486' is assigned to the CPU and the value 'CO\_processor' is assigned to EXTRAS.

## Table Calls

In the RESTRICTIONS section of a constraint, you can also call tables that you defined previously. This makes a constraint a powerful tool, since you can replace a large number of preconditions by using a table call in a constraint.

You can use tables in constraints to ensure the consistency of assigned values or to infer values. For restrictable characteristics, you can also use a table to dynamically restrict the allowed values of the characteristic. For example, you can use a table call to define the combinations of values for the casing and hard disk of a PC. If you refer to a table in a constraint, only the combinations of values defined in the table are valid.

## Settings Values in the Restrictions Section

In some cases, values can be inferred from your entries in the restrictions section without having to use the INFERENCES section. This is the case for equations where the left-hand side contains exactly one variable for a characteristic. The characteristic cannot be restrictable.



Voltage = Resistance \* Current

CPU = '486'

In the first example, the voltage of a circuit is inferred automatically from the equation without having to enter anything under INFERENCES. In the second example, the value '486' is set automatically for the CPU in the configuration.

**Constraints: Setting Values**

## Constraints: Setting Values

In the INFERENCES section of a constraint, you can define the characteristics for which the constraint sets values. Only enter characteristics here if you really want to use the constraint to set values. If you want to use the constraint to monitor the consistency of the configuration, leave out this section.



Do not enter the characteristic variable for which values are to be inferred in the constraint condition or in an IF condition. The inference must be technically possible. This means that the relevant data fields must be defined for the table calls or function calls, arithmetical equations must allow a solution for the variable, and so on.

If you want to use formulas to infer values, you do not need to enter a separate formula for each characteristic, as is the case with some other programming languages:

Voltage = Resistance \* Current

Resistance = Voltage/Current

Current = Voltage/Resistance

You only need to enter one equation in the RESTRICTIONS section of the constraint. In the INFERENCES section, you only enter the objects or variables you want to infer.



Either all the variables to be inferred or none of the variables to be inferred must be restrictable.

**OBJECTS:**

(300)Circuit

where V = Voltage; R = Resistance; C = Current

**RESTRICTIONS:**

V = R \* C

**INFERENCES:**

V, R, I

## System Performance

To improve system performance, we recommend that you only enter characteristic values to be inferred if you really need them inferred.

## Table Calls

You can use tables in constraints. You can use tables to check the consistency of assigned values, to infer values, or to restrict the allowed values of a characteristic. For more information, refer to [Tables in Constraints \[Page 191\]](#).

## Why Not Explanations

If you want to see why a constraint has not inferred a value for a specific characteristic or why a specific characteristic value has not triggered a constraint, you can use the explanation facility or trace functions on the characteristic value assignment screen to find more information (see [Explanation Functions for Value Assignment \[Page 243\]](#)).



**Constraints: Restricting the Allowed Values**

## Constraints: Restricting the Allowed Values

You can use the INFERENCES section of a constraint to restrict the allowed values of a restrictable characteristic, as well as to infer values, if:

- The characteristic of the class is defined as *Restrictable* (see [Restrictable Characteristics \[Page 146\]](#))
- Either all the variables to be inferred or none of the variables to be inferred must be restrictable.



Restricting the allowed values puts a heavy load on system performance.

If you use tables to restrict allowed values, the system load depends on the size of the table and the number of columns in the table for which values are inferred.

You have various options for restricting the allowed values.

### Restricting Allowed Values with Tables

You can refer to tables in a constraint to restrict the allowed values of a characteristic dynamically. You can define allowed combinations of values by maintaining the table. You can refer to the table in a constraint. The allowed values for characteristics can then be restricted according to the combinations of values maintained in the table.

For more information on working with tables, see [Tables in Constraints \[Page 191\]](#).

### Restricting Allowed Values with IN

You can also specify allowed values by using the expression **IN**:

**OBJECTS:**

PC IS\_A (300) PC

WHERE C = CASING;HD = HARD\_DISK.

**RESTRICTIONS:**

HD IN ('1275', '1620', '2000') IF C = 'Tower'.

**INFERENCES:**

HD.

### Restricting Allowed Values with a Linear Function

This type of restriction is only possible for numeric characteristics. In the RESTRICTIONS part of a constraint, you can enter a restriction for restrictable numeric characteristics in the form  $f(x) < 0$  (right-hand side constant value) such that  $f(x)$  is a linear expression in variable X. Any other comparison operator can be used in the place of '<'.



**RESTRICTIONS:**

5L - 20 > 0

**Constraints: Restricting the Allowed Values**

**Restricting Allowed Values with Comparisons**

You can restrict the allowed values of alphanumeric characteristics by comparing two alphanumeric characteristics.

**OBJECTS:**

PC is\_a (300) PC

**RESTRICTIONS:**

PC.COLOR\_1 = PC.COLOR\_2

**INFERENCES:**

PC.COLOR\_1

COLOR\_1 and COLOR\_2 are restrictable characteristics. The constraint assigns the values of characteristic COLOR\_2 to characteristic COLOR\_1.

## Creating a Dependency Net

## Creating a Dependency Net

### Use

Constraints are grouped together in dependency nets. For this reason, the variant configuration menu does not support a function for creating constraints directly. You always create a constraint within a dependency net.

### Procedure

1. From the variant configuration menu, choose *Dependency* → *Dependency net* → *Create*.
2. You see the initial screen. Enter a name for your dependency net.
3. You see the basic data screen.
  - You must enter a language-dependent description for your dependency net.
  - The default dependency net type is constraint net, because this is the only type of dependency net currently supported.
  - In the standard R/3 System, you see the status *'In preparation'* when you first create a dependency net. You cannot set the status to *'Released'* until you have created a language-dependent description.
4. If you select the *Allocations* pushbutton, you see a screen on which you can allocate constraints to the dependency net.
5. Enter a name for the constraint.
6. You see a dialog box, where you confirm the constraint name.
7. You see the basic data screen for creating a constraint.

On the basic data screen, you enter a language-dependent description for the constraint.

- You can choose *Extras* → *Documentation* to maintain a text describing the dependency net.

The documentation is divided into two sections: *Explanation* and *Documentation*.

In the *Explanation* section, you can enter a language-dependent explanatory text for the dependency. This text is displayed on the value assignment screen when you call the explanation component for a constraint.

In the *Documentation* section, you can save technical documentation, which is not displayed.



- a) A constraint has been violated when configuring a material, and the *Inconsistency* pushbutton appears.
- b) Choose *Inconsistency*.  
You see a dialog box containing the error message.
- c) Place the cursor on the error message and choose *Detail*.  
You see the *Explanation* and the source code of the constraint.



### Creating a Dependency Net

8. Choose *Goto* → *Dependency editor* to see the dependency editor. This is where you enter your source code.
9. Once you have entered your source code, you can use the *Dependency* → *Check* function to run a syntax check on your source code.

You can still save a constraint if the syntax contains errors. The status of the constraint is automatically set to '*Locked*' and you cannot use the constraint in configuration.

10. Save your source code. If the source code is finished, change the status of the constraint to '*Released*', so that you can use the constraint in configuration.
11. Once you have saved your constraint, you see the allocation screen again. Save the allocation of the constraint to the dependency net.



Check whether the status of the dependency net is also '*Released*'.

### Result

To allocate the dependency net to a configurable material, choose *Extras* → *Object dependencies* → *Allocations* from the configuration profile.

---

Changing a Dependency Net

## Changing a Dependency Net

### Procedure

To change a constraint, first you must enter the dependency net to which the constraint is allocated. You can make changes that are valid immediately or changes that become valid on a specific date in the future. If you want to make your change in the future, you must enter a change number with the valid-from date you require (refer to [Changing a Single Dependency \[Page 68\]](#)).

1. Choose *Dependency* → *Dependency net* → *Change*.
2. If you want your change to become valid on a specific date in the future, enter a valid change number.
3. Choose *Goto* → *Allocations*.

You can change the allocations to the dependency net.

4. If you want to make changes to a specific constraint, select the constraint you require.

If you choose *Environment* → *Change overview*, you can display the validity period of a constraint.

## Displaying a Dependency Net

To display a dependency net, choose *Dependency* → *Dependency net* → *Display*.

1. Enter the name of the dependency net to which the constraint is allocated.  
If you want to display the dependency net on a specific date, enter the valid-from date you require.
2. Choose *Goto* → *Allocations* from the basic data screen to display the constraints that are allocated to the dependency net.
3. You can select a specific constraint and display it.

## Networks in a Company

## Networks in a Company

This example of a network for a company illustrates how constraints are used and how they work. This example shows how you can use constraints to do configuration tasks over several levels, inferring values for a subordinate configurable material by configuring the superior configurable material.



This example is not intended to provide any kind of technical information for setting up a network. The only purpose of this example is to illustrate the use of constraints.

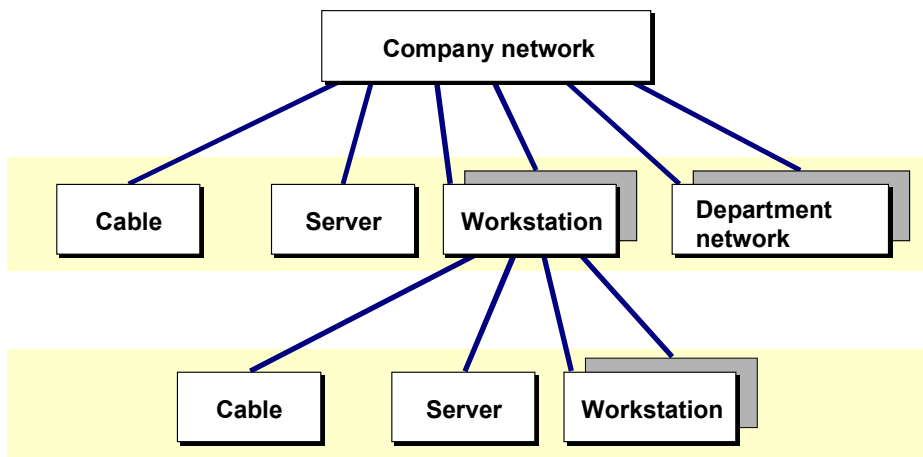
### Concept

A company network consists of different LAN types, operating systems, cables, and motherboards with different processors. Not all of these objects can be combined with each other. You can use object dependencies to model the interdependencies between these objects.

A LAN consists of cables, a server, and workstations. A workstation and a server contain a motherboard, an operating system, and a network card. If the LAN has a subordinate network, this network has its own server. You can use object dependencies to describe these interdependencies.

### Structure

The following graphic shows the structure of the network in this example:



The servers, the company network, the workstations, and the department network are all configurable materials:

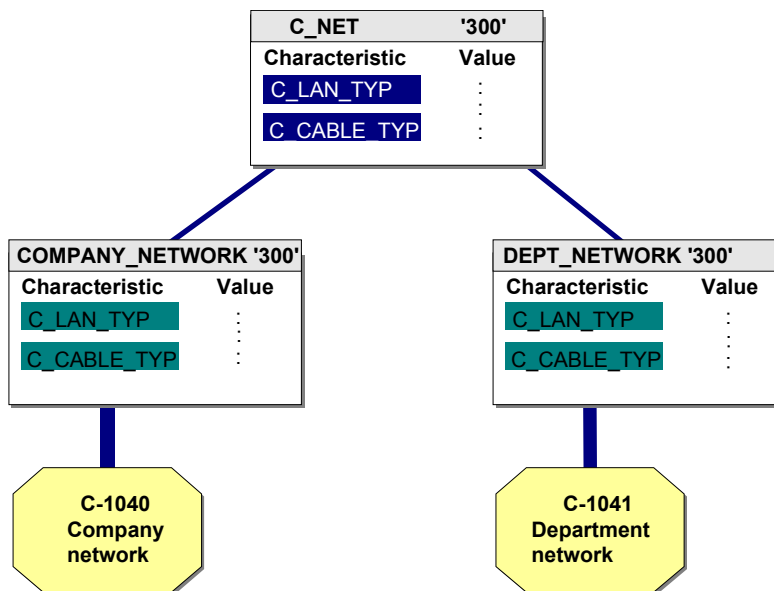
Material Number	Description	
C-1040	Company network	
C-1041	Department network	

Networks in a Company

C-1042	Workstation	
C-1043	Server	

Workstations and department networks occur twice in the BOM of the company network. However, since the department network is a self-sufficient network, you can also sell this component on its own. For this reason, you need a separate configuration profile for the department network.

A configuration profile links the company network to class 300 COMPANY\_NETWORK, and the department network to class 300 DEPT\_NETWORK. These two classes are assigned to superior class 300 C\_NET. The characteristics of class C\_NET are inherited by the two subordinate classes, which have no characteristics of their own.

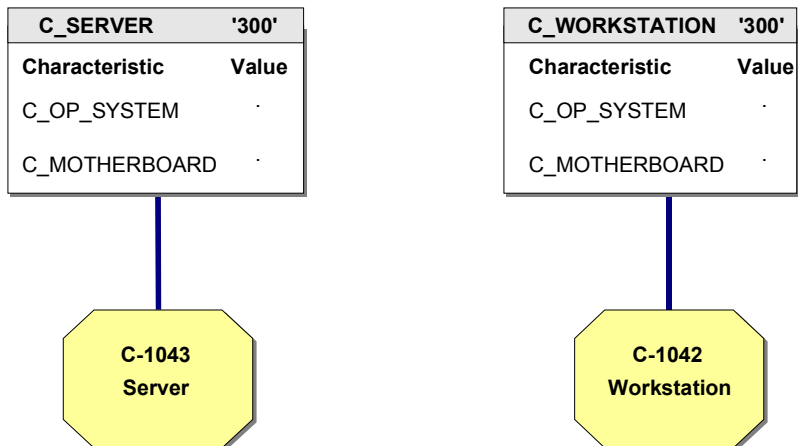


  = Characteristic in class

  = Inherited characteristic

Servers are assigned to class 300 C\_SERVER via the configuration profile, and workstations are assigned to class 300 C\_WORKSTATION.

## Networks in a Company



The classes have the following characteristics:

**C\_NET**

Characteristic	Value
C_LAN_TYPE	Ethernet Thinwire
	Ethernet Thickwire
	Token Ring
	Token Bus
	FDDI
Characteristic	Value
C_CABLE_TYPE	Twisted pair
	Shielded twisted pair
	Coax cable
	Glass fiber

**C\_SERVER and C\_WORKSTATION**

Characteristic	Value
C_OP_SYSTEM	OS1
	OS2
	OS3
Characteristic	Value
C_MOTHERBOARD	486 SX
	486 DX

	Pentium
	68020
	68030
	68040
	Sparc

## Interdependencies

The following interdependencies are to be described by using constraints:

1. Relationship between operating system and server
2. Relationship between operating system of server and operating system of workstation
3. Relationship between operating system of department server and operating system of company server
4. Relationship between LAN type and server processor
5. Relationship between cable type and LAN type

---

Relationship Between Operating System and Server

## Relationship Between Operating System and Server

This example shows how you can use a constraint to define that certain relationships are **false**. The constraint checks that such relationships do not occur in the configuration. If the relationship defined as false occurs in the configuration, you see an inconsistency message.

First, you want to define that a server cannot run with operating system OS2.

Dependency net: C\_Server

Dependency: No\_OS2\_as\_Server

Source code:

```
OBJECTS:
Server is_a (300) c_server

CONDITION:
server.c_operating_system = 'OS2'

RESTRICTIONS:
false
```

Under OBJECTS: you define that the constraint refers to a server. The server is referred to by class '300' C\_Server to which it is allocated.

Under CONDITION: you define that the constraint is only valid if the value **OS2** is selected for characteristic **c\_operating\_system**.

Under RESTRICTIONS: you define that the above condition always leads to an inconsistency. This means that operating system OS2 cannot be selected for the server.

You allocate the dependency net containing the constraint to both the company network and the department network. Since you can order the department network without ordering the company network, it is important to ensure the consistency of the department network, too.



## Relationship Between Operating System of Server and Operating System of Workstation

This example shows how a constraint refers to the characteristics of two different objects to ensure the consistency of the assigned values.

The operating system for workstations must be OS1 if the operating system of the server is OS1:

Dependency net: C\_net

Dependency: OS1\_workstation

Source code:

```
OBJECTS:
Server is_a (300) c_server
WHERE server_os = c_operating_system,
Workstation is_a (300) c_workstation
WHERE workstation_os = c_operating_system,
Net is_a (300) c_net

CONDITION:
part_of (server, net) and part_of (workstation, net) and
server_os = 'OS1'

RESTRICTIONS:
Workstation_os = server_os

INFERENCES:
Workstation_os
```

The constraint refers to the objects server, workstation, and network. These objects are referred to by entering the classes to which they are allocated. The constraint also contains the characteristics to be compared by the constraint.

The condition defined for the constraint is that server and workstation must be part of a network (refer to [Constraints: Entering Conditions \[Page 111\]](#)). Since class **c\_net** is a superior class of classes **company\_network** and **department\_network**, the statements in the constraint affect both the company network and the department networks. This means that you do not need to create two different constraints for the company network and the department networks.

Under RESTRICTIONS you define that the operating system of the workstation must be the same as the operating system of the server. Since you defined in the CONDITION section that the operating system of the server must be OS1, operating system OS1 must be inferred for the workstation.

Under INFERENCES you define that the operating system for workstations is to be inferred. This entry is for information only, because an equation under RESTRICTIONS automatically determines the values for the variable on the left-hand side of the equals sign (refer to [Restrictions \[Page 114\]](#)).

You link the dependency net containing this constraint to both the company network and the department network. This means that the constraint still takes effect if you sell a department network on its own.

---

Relationship Between Operating System of Server and Operating System of Workstation

## Relationship Between Company Network Server and Department Network Server

This constraint is based on the same principle as the previous constraint. However, this constraint is to be used for assigning values across 2 levels. It shows that the operating system of the department server is dependent on the operating system of the company server.

The server of the department network must run on OS1 if the server of the company network runs on OS1.

Dependency net: c\_company\_network

Dependency: OS1\_server

Source code:

```
OBJECTS:
Server1 is_a (300) c_server
WHERE server1_os = c_operating_system,
Server2 is_a (300) c_server
WHERE server2_os = c_operating_system,
company is_a (300) c_company_network,
department is_a (300) c_department_network.

CONDITION:
part_of (server1, company) and
part_of (department, company) and
part_of (server2, department) and
server1_os = 'OS1'.

RESTRICTIONS:
Server2_os = server1_os.
```

First, variables are defined for the relevant objects. Since we are talking about 2 servers here, two variables for servers must be defined. However, the two servers are the same material. Variables are also defined for a company network and a department network. The networks are referred to by entering the class to which they are assigned.

You define that Server1 belongs to the company network and a department network belongs to the company network. Server2 belongs to the department network. The operating system of the company server is OS1.

In the RESTRICTIONS section, you define that the operating system of the company server must be the same as the operating system of the department server. Since this condition infers that the operating system of the company server is OS1, the operating system of the department server must also be OS1.

In this example, you do not need to make any entries in the INFERENCES section because the equation in the RESTRICTIONS section automatically infers the values for the variable on the left-hand side (see [Constraints: Restrictions \[Page 114\]](#)).

---

Relationship Between Company Network Server and Department Network Server

## Relationship Between LAN Type and Server Processor

This example describes a constraint that is used only to check the consistency of the configuration, not to infer values. It shows the dependency between LAN type FDDI and the processor selected.

If the LAN type is FDDI, the motherboard of the server must contain either a Pentium, Motorola 68040, or Sparc 5 processor.

Dependency: FDDI\_Motherboards

Source code:

```
OBJECTS:
network is_a (300) c_net
WHERE Lan = c_lan_type,
Server is_a (300) c_server
WHERE Motherboard = c_motherboard

CONDITION:
Part_of (server, network) and network.Lan = 'FDDI'

RESTRICTIONS:
Motherboard in ('Pentium', '68040', 'Sparc')
```

The constraint refers to the objects network and server. Since the servers of both department networks and company networks are allocated to classes that are subordinate classes of class **c\_net**, the constraint uses class **c\_net** to refer to the servers of both department networks and company networks.

The condition defines that the server must be part of a network and the LAN type for the network must be FDDI. If this condition is true, the motherboard can only contain a Pentium, Motorola 68040, or Sparc 5 processor.

The constraint must be allocated to both the company network and the department network.

**Dependency: Cable Type and LAN Type****Dependency: Cable Type and LAN Type**

This example shows how tables can be used in constraints. The table in the example describes the following allowed combinations of values for characteristics C\_LAN\_TYPE and C\_CABLE\_TYPE:

C_LAN_TYPE	C_CABLE_TYPE
Ethernet Thinwire	Twisted pair
Ethernet Thinwire	Shielded twisted pair
Ethernet Thickwire	Coax cable
Token Ring	Twisted pair
Token Ring	Shielded twisted pair
Token Ring	Glass fiber
Token Bus	Coax cable
FDDI	Glass fiber



Tables used in dependencies can only contain single-value characteristics. The characteristics of the classes that are compared with the characteristics in the table can also be multiple-value characteristics or restrictable characteristics.

This table is used in the constraint to define which combinations of values are valid. If an invalid combination of values is assigned when you configure a material, the constraint triggers an inconsistency message.

**OBJECT:**

```
network is_a (300) c_net
```

**RESTRICTIONS:**

```
table cable_lan
(c_lan_type = network.c_lan_type,
c_cable_type = network.c_cable_type) .
```

The constraint refers to all networks, because class C\_NET is entered as an object. This class covers both company networks and department networks.

Under RESTRICTIONS, the characteristics of the table are compared with the characteristics of the class. In this example, the same characteristics are used in both the table and the class.

You can also use tables in constraints to infer values. To do this, you enter the characteristic whose value you want inferred by the table under INFERENCES. However, this is only possible if a unique value can be inferred. If you want to infer a value for the cable type, and the value 'Token Ring' is assigned to the LAN type, a unique value cannot be inferred for the cable type, because there are three possible values.

You can also use tables in constraints to restrict the allowed values of a characteristic. However, to do this you must create separate characteristics for the class and the table. The characteristics of the class must be defined as restrictable (refer to [Constraints: Inferring Values \[Page 116\]](#)).



## Actions (Obsolete)

## Actions (Obsolete)



Do **not** use actions if you can avoid it. You can almost always use procedures or constraints to do the same thing. All of the same expressions can be used in procedures. If you use actions **and** procedures, this can lead to serious system performance problems. For this reason, avoid using actions and replace existing actions with procedures or constraints when possible.

### Definition

You use actions to infer values for characteristics. You cannot overwrite values that are set by an action.

You can also use actions to change field values in BOM items or operations in routings.

You can assign actions to the following objects:

- The characteristic value that triggers the action
- The characteristic that triggers the action
- The configuration profile of the configurable object



It is easier to assign actions to the configuration profile, so that they are all in one place.

- BOM items – to change the quantity, for example
- Operations in task lists – to change the standard values, for example

### Use

- If you use an action to infer a value for a characteristic, you enter the variable `$SELF` before the characteristic.
- You cannot use the following expressions in actions:
  - `NOT SPECIFIED`
  - `NOT TYPE_OF`
  - `<multiple-value characteristic> NE <value>`

#### See also:

[Changing Master Data with Dependencies \[Page 96\]](#)



## Inferring a Characteristic Value with Actions (Example)

### Use

Configurable material BIKE has the following characteristics:

Characteristic	Values	Actions
MODEL	<b>Racing</b>	SADDLE = 'Leather'
	Standard	
	Mountain	
	Tandem	
SADDLE	Plastic	
	<b>Leather</b>	

If the value 'Racing' is set for MODEL, the value 'Leather' is automatically set for SADDLE, because racing bikes are always supplied with a leather saddle.

### Procedure

1. Create an action.
2. Enter the following source code in the dependency editor:  
`$SELF.SADDLE = 'Leather' if MODEL = 'Racing'`
3. Allocate the action to the configuration profile of material BIKE.

### Result

If the value 'Racing' is set for MODEL when you configure the BIKE, the value 'Leather' is automatically set for SADDLE.

#### See also:

[Allocating Object Dependencies \[Page 92\]](#)

---

**Where-Used List for Dependencies**

## Where-Used List for Dependencies

You can produce a list of objects where a dependency is used. To do this, choose *Dependency* → *Where-used list* from the variant configuration menu.

1. Enter the name of the dependency for which you want to display a where-used list. Since you can process dependencies using engineering change management, you can enter a date on which you want to produce the where-used list.
2. You see a list of objects in which the dependency is in use:
  - In configurable materials
  - In BOM items
  - In characteristics
  - In characteristic values
3. To display an object from the list, choose *Detail*.

## Dependency List

You can produce a list of dependencies according to the following selection criteria:

- Date on which dependency was created
- Dependency type
- Dependency group
- Dependency name
- User
- Status
- Change number

To produce a list of object dependencies, proceed as follows:

1. From the variant configuration menu, choose *Dependency* → *Dependency list*.  
Enter the criteria according to which you want to produce your list.  
You can also produce the list for a certain date.
2. Execute the function. You see a list of dependencies that match your selection criteria.  
To display further information and the source code of a dependency, choose *Details*.  
You can also display a where-used list for a dependency in the list.

**Declarative Dependencies**

## Declarative Dependencies

### Definition

In declarative dependencies, the point in time when the dependency is processed and the sequence in which the dependencies are processed are not relevant to the result. The result of a declarative dependency can be explained logically. The dependency describes a rule that must always apply. This contrasts with procedural dependencies, where the result depends on the processing sequence and the point in time when the dependency is processed.

As described in [Procedures \[Page 81\]](#), some expressions are in themselves non-declarative. For this reason, these expressions cannot be used in declarative object dependencies:

- NOT SPECIFIED
- NOT TYPE\_OF
- <multiple-value characteristic> NE <value>

These expressions assume that you deliberately assigned no value to some characteristics. However, during the configuration process, it is not clear whether a piece of information is not given deliberately or whether a value will be assigned to a characteristic at a later point in time. For this reason, missing information cannot be used to define a rule for declarative dependencies.

You can use these expressions in preconditions and selection conditions. Since preconditions and selection conditions are not processed until the end of the configuration process, you can assume that values have been assigned to all the characteristics that are intended to have values, and that any missing information is deliberately not given. However, preconditions and selection conditions then no longer count as declarative object dependencies.

When the not equals (NE) expression is processed for multiple-value characteristics, there is a difference between constraints and actions. In constraints, 'not equals' can be expressed for multiple-value characteristics, because individual values are processed in constraints. In actions, however, the sum of the selected values is processed. Since it is not clear whether it is deliberate that no value was selected for a multiple-value characteristic or whether a value will be assigned at a later point in time, this expression cannot be processed in actions.



Declarative dependencies are easier to trace than procedural dependencies, because the point in time when the dependency is processed and the sequence in which the dependencies are processed are not relevant.

The following dependency types are declarative dependencies:

- Constraints
- Actions
- Preconditions (provided that they contain no non-declarative expressions)
- Selection conditions (provided that they contain no non-declarative expressions)

The following dependency types are procedural object dependencies:

- Procedures

**Declarative Dependencies**

- Preconditions (only if they contain non-declarative expressions)
- Selection conditions (only if they contain non-declarative expressions)

## Value Assignment Attribute for Characteristics

# Value Assignment Attribute for Characteristics

In characteristics maintenance functions, you define whether a characteristic is:

- Single-value
- Multiple-value
- Restrictable

If a characteristic is a single-value characteristic, the characteristic can only have one value. For example, you can only select one model for a car.

If you allow multiple values for a characteristic, the characteristic can have more than one value at a time. For example, you can select several different values for characteristic EXTRAS of a car.

A restrictable characteristic is in effect a single-value characteristic. However, the allowed values for a restrictable characteristic can be restricted dynamically during the configuration process. For example, if the allowed values of a characteristic are restricted by dependencies to certain values, these values can be temporarily assigned to the characteristic during the configuration process, but the configuration is not complete until you select one specific value.



The value assignment attribute of a characteristic is a global attribute that applies to the characteristic wherever it is used.

Most characteristics used in dependencies are single-value. Check the settings in Customizing for the *Classification System*. By choosing *Characteristics* → *Define default Settings*, you can enter default values. Please ensure that the *Multiple values* indicator is **not** selected, because this would mean that the characteristics you create have *Multiple values* allowed as a default setting.

Multiple-value characteristics and restrictable characteristics are difficult to process in object dependencies and should be used with caution. For information on how multiple-value characteristics and restrictable characteristics are processed, refer to:

- [Single-Value Characteristics \[Page 143\]](#)
- [Restrictable Characteristics \[Page 146\]](#)
- [Multiple-Value Characteristics \[Page 144\]](#)

## Single-Value Characteristics

You can use dependencies to assign exactly one value to a single-value characteristic. If more than one value is assigned, an inconsistency occurs.

You refer to a single-value characteristic in dependencies as follows:

- The condition SPECIFIED MODEL is fulfilled if exactly one value is assigned to characteristic MODEL. If no value is assigned to the characteristic, the condition is not fulfilled.
- In the equation MODEL = 'Racing', only the value 'Racing' can be assigned to the characteristic.  

If no value is assigned, and the dependency is a precondition, the condition is not violated, so the precondition is processed. If no value is assigned, and the dependency is a selection condition, the condition is violated, so the precondition is not processed (refer to [Preconditions \[Page 73\]](#) and [Selection Conditions \[Page 77\]](#)).
- The condition MODEL IN ('Racing', 'Trekking') is fulfilled if one of these values is assigned to the characteristic. If a value is assigned that is not one of the allowed values, the dependency is not processed. If no value is assigned, and the dependency is a precondition, the condition is not violated, so the precondition is processed. If no value is assigned, and the dependency is a selection condition, the condition is violated, so the selection condition is not processed.

## Multiple-Value Characteristics

### Multiple-Value Characteristics

Multiple-value characteristics can have several values at a time. Multiple-value characteristics often cause problems in dependencies, because it is not always clear how values are to be compared with each other.

The comparison of two multiple-value characteristics is processed differently in different dependency types.

#### Processing in Constraints

In constraints, you can compare two multiple-value characteristics with each other, because the individual values are compared, rather than the quantity of characteristic values. This means that first value 'A' is compared, then value B, and so on. If the CONDITION section of a constraint contains a comparison of two multiple-value characteristics, the constraint is only processed for pairs of values in the two characteristics.

If the same equation is entered in the RESTRICTIONS section of a constraint, each individual value of A is compared to the individual values of B. Since comparisons in the RESTRICTIONS section of a constraint tend to automatically infer values for the characteristics on the left-hand side of the equation (refer to [Constraints: Inferring Values \[Page 116\]](#)), this would mean that the values of B are assigned to A.



Since equations in the RESTRICTIONS section of a constraint always automatically lead to inference of values, you cannot use them to check the configuration.

The equation `A.Extras <> 'Central_locking'` in a constraint condition means that the constraint can be processed if any value that is not 'Central\_locking' (for example, 'ABS' and 'Sunroof') is assigned to characteristic EXTRAS. If you enter this equation in the restrictions section, an inconsistency occurs if the value 'Central\_locking' is selected as one of the extras.

#### Processing in Preconditions, Selection Conditions, and the Conditional Section of Procedures and Actions

In these dependency types, the quantity of values is compared to each other. The comparison of two multiple-value characteristics is not possible in preconditions, selection conditions, procedures, and actions, because the meaning of the comparison is ambiguous.

For example, if car A has the extras ABS and Sunroof, and another car B has the extras ABS, Sunroof, and Central locking, the following equation can have different meanings:

$$A.Extras = B.Extras$$

The possibilities are:

1. The quantity of all values assigned to A and B must be the same. This condition would not be fulfilled for A and B, because B has the additional value 'Central locking'.
2. At least one common value must exist. This condition would be fulfilled for A and B.
3. A must be given the same extras as B (inference). The values of A form a subset of the values of B.

Inequality can also have different meanings for multiple-value characteristics:

$$A.Extras <> 'Central\_locking'$$



## Multiple-Value Characteristics

1. You cannot select the value 'Central\_locking' for EXTRAS.
2. At least one value other than 'Central\_locking' must exist.

In conditions, a characteristic must be compared with explicit values:

```
$ROOT.EXTRAS = 'ABS'
```

```
$ROOT.EXTRAS <> 'ABS'
```

For the condition to be fulfilled in the first case, the value 'ABS' must be assigned to the root object. However, other values can also be assigned to characteristic EXTRAS.

For the condition to be fulfilled in the second case, the value 'ABS' must not be assigned to characteristic EXTRAS.

For more information on the syntax of comparisons between multiple-value characteristics in dependencies, see [Comparisons \[Page 163\]](#).

### Processing in the Inferences Section of Procedures and Actions

The INFERENCES section of actions and procedures can contain statements such as:

```
$SELF.EXTRAS = $ROOT.EXTRAS
```

This is allowed because the values of the root object \$ROOT are inferred for the object currently being processed – \$SELF.

### IN

The following comparison cannot be used in single dependencies:

```
<multiple-value characteristic> IN ('Value1', Value2')
```

The condition is ambiguous. It could mean either that at least one of the values must be entered for the characteristic or that all the values in the set must be entered.

You can enter this condition in a constraint, because a constraint reads values individually.

However, you cannot use IN for multiple-value characteristics to define that a value must be set:

```
'ABS' IN EXTRAS
```

### SPECIFIED

The following condition is fulfilled if one or more values is assigned to the characteristic:

```
SPECIFIED & <multiple-value characteristic>
```

## Restrictable Characteristics

## Restrictable Characteristics

### Use

Restrictable characteristics let you manipulate the allowed values of characteristics. If the allowed values of a characteristic depend on the values assigned to other characteristics, you can use restrictable characteristics to ensure that you only see the values that are really allowed, so that your configuration is consistent.



The frame height is dependent on whether it is a men's or a women's bicycle. You can restrict the range of values for the characteristic FRAME\_HEIGHT to the allowed values.

MODEL	TYPE	FRAME_HEIGHT
Trekking	Women	48, 53
Trekking	Men	48, 53
Mountain	Women	46, 50, 53, 57
Mountain	Men	50, 54, 58
Racing	Women	46, 49, 53, 57
Racing	Men	49, 53, 57, 61, 64



Restrictable characteristics can have an adverse effect on system performance. For this reason, use them with caution.

As an alternative to restrictable characteristics, you can use preconditions to restrict the allowed values of a characteristic.

### Prerequisites

The characteristic has the attribute *Restrictable* in its formatting data.

The configuration editor does not show restrictable characteristics as different to other characteristics. For this reason, it is useful to give restrictable characteristics a special name or description to show that they are restrictable.

### Features

You can only restrict restrictable characteristics by using constraints.

Restrictable characteristics are handled in the same way as single-value characteristics in all other dependency types:

### Restrictable Characteristics

- If characteristic MODEL is restrictable, the condition SPECIFIED MODEL is only fulfilled if the allowed values for characteristic MODEL have been restricted to exactly one value.
- In the equation MODEL = 'Racing', only the value 'Racing' can be assigned to the characteristic. The dependency is processed when the allowed values are restricted to 'Racing'.
- The condition MODEL IN ('Racing', 'Trekking') is fulfilled if one of these values is assigned to the characteristic.

## Restricting a Characteristic

# Restricting a Characteristic

## Use

You can only restrict restrictable characteristics by using constraints.

Under RESTRICTIONS: you can restrict a characteristic as follows:

- Call a variant table that contains the valid combinations of values.  
Refer to [Restricting Characteristics with a Variant Table \[Page 149\]](#)
- Operator 'IN'

Operator 'IN'

<Characteristic> IN ('Value1', Value2, 'Value3',...)

- Compare 2 alphanumeric characteristics that are both restrictable

Characteristic\_1 = Characteristic\_2

As a result of the comparison, the allowed values for Characteristic\_2 are assigned to Characteristic\_1.

- Linear function

For numeric characteristics that are restrictable, you can use a linear function in the form

$F(x) < 0$ .

- A constant value is given on the right-hand side.
- $F(x)$  is a linear expression with the variable  $X$ .
- Any other comparison operator can be used in the place of '<'.



$5x - 20 > 0$

## Restricting Characteristics with a Variant Table

### Purpose

You can use variant tables, containing valid combinations of values, to restrict the allowed values of a characteristic. You refer to the variant table in a constraint.



You need not define value assignment alternatives to restrict values in the variant table.

### Prerequisites

The characteristics (of the class) whose allowed values you want to restrict must be defined as restrictable.

Because dependencies does not allow restrictable characteristics, you must not use restrictable characteristics in the table. Therefore, define several single-value characteristics in the table that have identical formats and values as the characteristics of the class to which you are allocating them.

### Process Flow

1. You create a variant table, entering the characteristics with the valid combinations of values.  
In each table line, enter at least one valid value for the characteristic you want to restrict. If a table line does not contain a value for the characteristic, the value set inferred during configuration is empty, which triggers an inconsistency.
2. Create a constraint.  
In the RESTRICTIONS section, you link the characteristics of the class to the characteristics of the table.  
In the INFERENCES section, you enter the characteristic whose allowed values you want to restrict.
3. Allocate the dependency net with the constraint to the configuration profile of the material.

### Result

When you configure the material, only the valid values are displayed for the restrictable characteristic.

You cannot manually extend the restricted allowed values of the characteristic later. To extend the allowed values again, you must delete the characteristic value that restricted them.

To display the hidden values, choose *View -> Scope* and select *Excluded*.

**Restricting a Characteristic with a Table: Example****Restricting a Characteristic with a Table: Example**

Configurable material BIKE has characteristics MODEL, TYPE and FRAME\_HEIGHT.

MODEL	TYPE	FRAME_HEIGHT
Trekking	Women	48
Trekking	Women	53
Trekking	Men	48
Trekking	Men	53
Mountain	Women	46
Mountain	Women	50
Mountain	Women	53
Mountain	Women	57
Mountain	Men	50
Mountain	Men	54
Mountain	Men	58
Racing	Women	46
Racing	Women	49
Racing	Women	53
Racing	Women	57
Racing	Men	49
Racing	Men	53
Racing	Men	57
Racing	Men	61
Racing	Men	64

Characteristic FRAME\_HEIGHT is restrictable. Characteristics MODEL and TYPE are single-value. You want the system to control the allowed values of characteristic FRAME\_HEIGHT.

**Procedure**

1. Create a single-value characteristic, T\_FRAME\_HEIGHT, especially for the table.



Tables that you specify in constraints must contain single-value characteristics only. This is why you need to define a characteristic especially for the table.

### Restricting a Characteristic with a Table: Example

2. Create table T\_FH, and enter the values combinations above. You need not define value assignment alternatives to restrict values.
3. Create dependency net CN\_BIKE and assign constraint CS\_FRAME\_HEIGHT to this dependency net.
4. Enter the following source code in the constraint:

```
OBJECTS:
BIKE IS_A (300) BIKE where MOD = MODEL; FH = FRAME_HEIGHT.
```

```
RESTRICTIONS:
TABLE T_FH
(MODEL = MOD, TYPE = TY, T_FRAME_HEIGHT = FH) .
```

```
INFERENCES:
FH.
```

5. Allocate the dependency net to the configuration profile of material BIKE.

## Result

On the value assignment screen, the allowed values of characteristic FRAME\_HEIGHT are restricted, depending on the values assigned to characteristics MODEL and TYPE.

---

**Restricting a Characteristic with IN**

## Restricting a Characteristic with IN

Configurable material BIKE has characteristics FRAME and COLOR. Characteristic COLOR has the values 'Red', 'Green', 'Blue', and 'Yellow'.

The values for characteristic COLOR are restricted to 'Green' and 'Yellow' if you choose an aluminum FRAME.

Characteristic COLOR is restrictable. Characteristic FRAME is single-value.

### Procedure

1. Create dependency net CN\_BIKE. If you have already created this dependency net, change it.
2. Allocate constraint CS\_FRAMECOLOR to the dependency net.
3. Enter the following source code in the constraint:

```
OBJECTS:
BIKE IS_A (300) BIKE where FR = FRAME; COL = COLOR

RESTRICTIONS:
COL IN ('Red', 'Yellow') IF FR = 'Aluminum'

INFERENCES:
COL.
```

4. Allocate the dependency net to the configuration profile of material BIKE.

### Result

If you set value 'Aluminum' for characteristic FRAME, you only see the values 'Yellow' and 'Green' for characteristic COLOR.



## Assigning Values to Restrictable Characteristics

1. Display the possible entries to see the values of the characteristic.
2. Select a value for the characteristic.
3. Confirm your entry.
4. The remaining values of the characteristic are no longer displayed.



To display all values, choose *View* → *Scope* and select *Excluded*. The excluded values are displayed but cannot be selected.

5. To change the value assigned to the characteristic, first delete the current value.
6. Confirm the changed value.
7. You see all the allowed values for the characteristic and can select another value.

## Using Dependencies to Change how Characteristics are Displayed

## Using Dependencies to Change how Characteristics are Displayed

### Purpose

You define the settings for displaying a characteristic and which fields are available for entry in characteristics maintenance. These settings apply to the characteristic wherever it is used.

However, if the way you want a characteristic displayed or the fields you want available for entry depend on the configuration environment, you now have the option of using dependencies to control these settings.

The different display options that you can define with dependencies are defined in structure SCREEN\_DEP.

- Hidden (INVISIBLE)  
The characteristic is hidden by dependencies.
- Ready for input (INPUT)  
A characteristic that was defined as *No entry can be made* in a characteristics maintenance function is available for entry.
- Not ready for input (NO\_INPUT)  
A characteristic cannot have a value assigned.
- Reset (RESET)  
Entries made by dependencies in structure SCREEN\_DEP are deleted.



Changes made with structure SCREEN\_DEP only apply to variant configuration, not to classification.

### Process Flow

1. Create a characteristic with a reference to structure SCREEN\_DEP.  
Enter a field in the structure.
2. Create a dependency that refers to the reference characteristic.  
Instead of a value, you assign a characteristic for the display option to the reference characteristic.
3. Assign this dependency to the configuration profile of the configurable material, or the characteristic.



#### Example Process Flow

- a) You create reference characteristic INVISIBLE with a reference to structure SCREEN\_DEP, field INVISIBLE.
- b) Create a procedure with the following source code:

### Using Dependencies to Change how Characteristics are Displayed

`$SELF.NO_DISPLAY = 'SADDLE' IF MODEL = 'STANDARD'`

In the procedure, you assign characteristic SADDLE to the characteristic. The condition is that characteristic MODEL must have the value 'Standard'.

- c) Assign the procedure to either characteristic SADDLE or the configuration profile of the material.
- d) If characteristic MODEL has another value, you do not see characteristic SADDLE.

**See also:**

[Creating a Reference Characteristic \[Ext.\]](#)

[Reference Characteristics \[Ext.\]](#)

## Dependency Syntax: General Rules

### Operators

Operators	Use in Dependencies
AND	Two statements that are both either true or not true are linked with AND. <code>Length = 300 and Width = 200</code>
OR	Two statements of which at least one is either true or not true are linked with OR. <code>Color = 'red' or Basic_material = 'wood'</code>
NOT	You can negate one or more expressions by using NOT. <code>NOT (Color = 'blue')</code> <code>NOT (Color = 'red' and Basic_material = 'wood')</code>
IF	Conditions in actions and procedures start with IF. <code>Color = 'red' if Model = 'A'</code>

### Special Features

Lists	In lists, the individual elements are always separated by commas. <code>COLOR = 'RED' IF MODEL = 'A',</code> <code>COLOR = 'BLUE' IF MODEL = 'B',</code> <code>COLOR = 'GREEN' IF MODEL = 'C',</code>
Case sensitivity	In characteristic names, object variables, and operators, there is no distinction between upper case and lower case letters.

### Concatenation

LC	All letters are converted to lower case. <code>Leather_saddle = LC('Alpha')</code> <code>= 'a'</code>
UC	All letters are converted to upper case. <code>Leather_saddle = UC('Alpha')</code> <code>= 'A'</code>  This function is important if the assigned characteristic does not allow lower case, but the assigned expression may contain lower-case letters.
	The string is cut off at the maximum number of 30 characters. <code>Leather_saddle = Alpha    Beta</code> <code>= 'AB'</code>

## List of Built-In Conditions

Expression	Implied Condition
SPECIFIED	Characteristic has a value: <b>SPECIFIED COLOR</b> See <a href="#">Built-In Condition SPECIFIED [Page 166]</a>
IN	One of these values must be set: <b>COLOR IN ('red', 'green', 'blue')</b> See <a href="#">Built-In Condition IN [Page 167]</a>
TYPE_OF	Only for certain objects: <b>TYPE_OF (\$ROOT, (Material) (300) (NR = 'U91'))</b> See <a href="#">Built-In Condition TYPE_OF [Page 168]</a>
PART_OF	The object is a component of a BOM (only in constraints). See <a href="#">Constraints: Entering Conditions [Page 111]</a>
SUBPART_OF	The object is a component of an assembly that is part of the BOM of a configurable material (only in constraints).

## Entering Characteristics and Characteristic Values

# Entering Characteristics and Characteristic Values

## Characteristics

You enter the language-independent name of the characteristic. No distinction is made between upper and lower case letters in characteristic names.

The following restrictions apply when using characteristics in dependencies:

- No characteristics with data types TIME, DATE, or CURR (currency)
- No characteristics with a user-defined data type, because dependencies can only read single characteristics.

## Character Values

- Character (alphanumeric) values are in single quote marks:  
`COLOR = 'RED'`
- The use of upper and lower case letters in character strings in quote marks depends on the settings made in characteristics maintenance.
  - If the characteristic values are not case sensitive, the values are automatically converted to upper case letters:  
`color = 'red'` is converted internally to `COLOR = 'RED'`
  - If a characteristic value is case sensitive, you must use upper and lower case letters as maintained in the characteristic.

## Numeric Values

- Numeric values are not in single quote marks:  
`Width = 15`
- You do not enter the unit.
- In figures with decimal places, you always use a period as a decimal point, never a comma:  
`Width = 2.34`
- Plus or minus signs go immediately before the value.  
`-100`
- A valid exponent is introduced by the character **E** (case is not relevant), and contains a plus or minus sign (+ or -) and a string of no more than two figures. An exponent cannot contain blank characters. The exponent itself must immediately follow the figure before it (without a blank character in between):  
`314E-2`  
`0.314E+01`



The system checks alphanumeric values to see if they exist in the characteristic.

There is no check for numeric values.



## Using Arithmetic Operations

## Using Arithmetic Operations

## Basic Arithmetic Operations:

You can use the following basic arithmetic operations in dependencies to perform calculations:

- + Addition
- Subtraction
- / Division
- \* Multiplication



`CHAR_WIDTH = CHAR_LENGTH / 4`

## Standard Functions

You can use the following standard functions:

sin	Sine function
cos	Cosine function
tan	Tangent function
exp	Exponent for base e
ln	Natural logarithm
abs	Absolute amount
sqrt	Square root
log10	Logarithm for base 10
arcsin	Arc sine (inverse function of sin)
arccos	Arc cosine (inverse function of cos)
arctan	Arc tangent (inverse function of tan)
sign	Sign (plus or minus sign) of x
frac	Decimal part of x



`SIN (2*3.14 * ANGLE / 360)`

No distinction is made between upper and lower case letters in functions. Function names are automatically converted to upper case letters.

## Precision and Rounding for Numeric Characteristics

You can use the following commands to round a value calculated by dependencies:

Function	Description	X = 3.1	X = -3.1
ceil	Lowest integer that is not less than x	+4	-3
trunc	Integer part of x	+3	-3



Using Arithmetic Operations

floor	Highest integer that is not greater than x	+3	-4
-------	--	----	----



Use in an action:

```
$SELF.CHAR_A = FLOOR ($SELF.CHAR_B + $SELF.CHAR_C + 0.5)
```

The action calculates the sum of characteristics B and C and rounds the result to an integer.

**You see the rounded value on the screen.** However, if the value is used in subsequent calculations, the **exact** value is used, not the rounded value.

## Entering Intervals

## Entering Intervals

You can enter intervals in round parentheses and in square parentheses. However, round parentheses can be ambiguous, because (1-5) could be either an interval or an arithmetic operation with the result -4.

You can express upper and lower limits in intervals using the following comparison operators:

### Upper Limit

Option 1	Option 2	Option 3	Description
>		GT	Greater than
>=	=>	GE	Greater than or equal to

### Lower Limit

Option 1	Option 2	Option 3	Description
<		LT	Less than
<=	=<	LE	Less than or equal to

## Syntax for Entering Intervals

Use the expression IN to enter intervals:

**LENGTH IN (5 - < 10)**

from 10 to 5, such that 10 is part of the interval but 5 is not

**LENGTH IN (5 - 10)**

from 5 to 10, such that 5 and 10 are part of the interval

**LENGTH IN (> 5 - < 10)**

from 5 to 10, such that neither 5 nor 10 is part of the interval

**LENGTH IN (> 5 - 10)**

from 5 to 10, such that 10 is part of the interval but 5 is not

**LENGTH IN (5 - 10, >20, 40)**

All lengths from 5 to 10, greater than 20, or 40

## Entering Comparisons

You can use the following comparison operators:

Option 1	Option 2	Option 3	Description
<		<b>LT</b>	Less than
<=	=<	<b>LE</b>	Less than or equal to
=		<b>EQ</b>	Equal to
>		<b>GT</b>	Greater than
>=	=>	<b>GE</b>	Greater than or equal to
><	<>	<b>NE</b>	Not equal to

For example, you can make the following comparisons:

**Length = 5**

**Length < 5**

**Length <= 5**

**Length > 10**

**Length >= 10**

## Object Variables

## Object Variables

You use object variables in a multi-level configuration, to describe configurable materials in a configuration structure.

You can use the following object variables:

- \$ROOT is the highest-level configurable material in a configuration
- \$SELF is the material to which the dependency is allocated

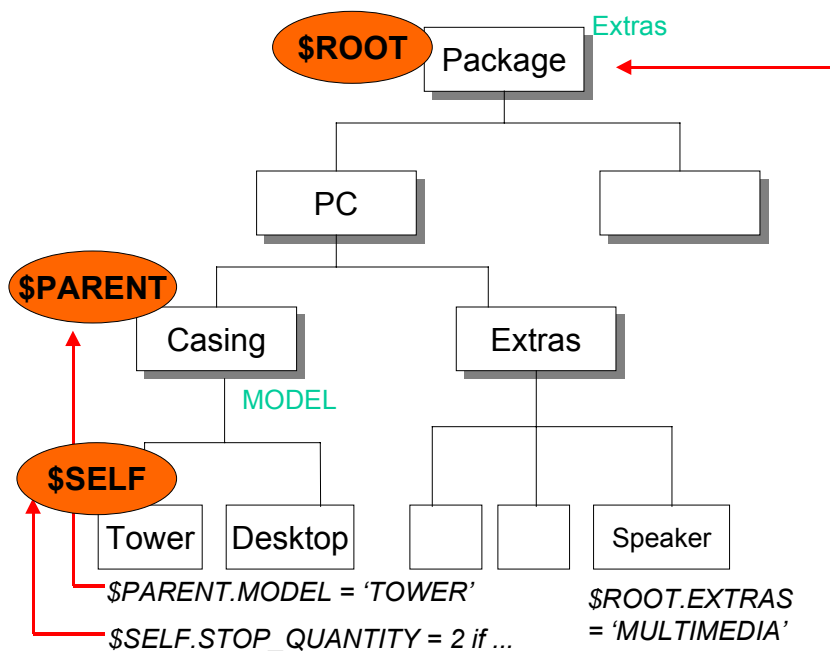
The material to which an action or procedure is allocated is the same material for which values are inferred by the action or procedure. To make it clear that actions and procedures only infer values for materials to which they are allocated, the characteristics are preceded by the variable \$SELF.

In dependencies that are allocated directly to the header material, \$SELF and \$ROOT both mean the header material.

- \$PARENT refers to the object immediately above \$SELF in a multi-level configuration.

The variable \$PARENT has no meaning for the header material, because the header material has no superior materials.

In dependencies that are allocated to BOM items, \$PARENT refers to the configurable material for which the BOM was created, whereas \$SELF refers to the material of the BOM item.





There is no variable for describing objects between the configurable material and the immediately superior material. This means that material PC cannot be accessed from materials TOWER, DESKTOP, and SPEAKER.

However, in a multi-level, interactive configuration \$ROOT may refer to two different objects:

- If the BOM is exploded in production, \$ROOT always refers to the material that transfers requirements.
- If the BOM is relevant to sales and distribution, \$ROOT always refers to the header material.

**Built-In Condition SPECIFIED**

## Built-In Condition SPECIFIED

### Use

You use the expression SPECIFIED in preconditions, selection conditions, or the conditional section of actions and procedures, to define that the condition is fulfilled if a characteristic has any value assigned to it. The specific value assigned is not relevant.



**SPECIFIED SPECIAL\_PAINT**

**SPECIAL\_PAINT SPECIFIED**

If you use a condition like this as, for example, a selection condition for an operation in a task list, the operation is included in the task list explosion if a value is assigned to characteristic SPECIAL\_PAINT when the condition is processed. It does not matter which value you assign to the characteristic.

The negative form NOT SPECIFIED always goes **before** the characteristic variable.



**NOT SPECIFIED SPECIAL\_PAINT**

### Restrictions

- You cannot use NOT SPECIFIED in actions and constraints.

## Built-In Condition IN

### Use

You can use the operator IN to define lists of values that have an OR relationship to each other.



```
COLOR IN ('red', 'yellow', 'green')
```

If you use a condition like this as, for example, a precondition for a characteristic, the characteristic appears if the value assigned to characteristic COLOR is either 'red', 'yellow', or 'green', or if no value is assigned to characteristic COLOR.

```
LENGTH IN (200, 300, 350)
```

If you use a condition like this as, for example, a selection condition for a BOM item, the item is included in the BOM explosion if the value assigned to characteristic POWER is 200, 300, or 350.

### Restrictions

Value nodes in value hierarchies are not supported in dependencies.

The expression **Country IN 'Europe'**, where **Europe** is a value node with subordinate values, is not allowed in dependencies.

**Built-In Condition TYPE\_OF**

## Built-In Condition TYPE\_OF

### Use

You use TYPE\_OF to define that the condition only applies to one object.

You use this condition if the dependency is dependent on its environment, for example, dependent on being used for a specific BOM header material.



```
TYPE_OF($ROOT, (Material) (300) (NR='U91'))
```

If you use a condition like this as, for example, a selection condition for a BOM item, the item is included in the BOM explosion if the \$ROOT object (header material) has material number 'U91'.

### Restrictions

- TYPE\_OF and NOT TYPE\_OF cannot be used in constraints.
- NOT TYPE\_OF cannot be used in actions.



## Variant Tables

### Purpose

Tables are used to store combinations of values for different characteristics – for example, you can only have a gray car interior if the car's paintwork is blue.

Tables are an aid to entry for dependencies. You enter the table in a dependency. The system uses the table to infer and check values. If the interdependencies between characteristics change, you change the table, not the dependencies.

### Integration

You can use tables in selection conditions, preconditions, actions, procedures, and constraints.

- In selection conditions, preconditions, constraints, and the conditional part of actions and procedures, tables are used to check the consistency of the values entered.
- In actions, procedures, and constraints, you can use tables to infer values.
- In constraints, you can use tables to restrict the allowed values for a characteristic (see [Restricting Characteristics with a Variant Table \[Page 149\]](#)).

### Features

There are two steps to creating a table:

1. You define the table structure.

You enter the characteristics that you want in the table and define value assignment alternatives.

The characteristics in the class can be identical to those in the table. However, only use single-value characteristics in tables. If the characteristics of the class are multiple-value, create single-value characteristics especially for the table. However, the characteristics must have the same format and values.

2. You maintain the table entries.

### Table Call in Dependencies

The characteristics of the table are compared with the characteristics of the class in dependencies.

```
TABLE <table name>  
  
(<characteristic of table> = <characteristic of class>,  
<characteristic of table> = <characteristic of class>)
```

### Link to a Database Table

You can link variant tables to database tables to provide new capabilities for accessing tables when processing data. This considerably improves system performance.

For more information, see [Link to a Database Table \[Page 178\]](#).

## Variant Tables

### Runtime Objects

Runtime objects are generated for all variant tables, and are updated when the variant tables are changed. Runtime objects contain the tables, table lines, and table entries in compressed form. This improves access to data.

### Engineering Change Management

Table contents can be maintained with engineering change management, both with a valid-from date and with effectivity.



Variant tables that have been maintained with engineering change management cannot be physically deleted from the database. Once their validity has expired, they only disappear logically, because engineering change management prevents them from being displayed.

You cannot process a table structure with engineering change management.

### Restrictions

To process tables in dependencies:

- Only use **single-value characteristics** in tables
- Do not use **intervals** in tables
- Only enter **one value** per table cell (otherwise inconsistencies occur when using the Sales Configuration Engine) If you enter more than one value in a cell, you can get the system to multiply out the values by choosing *Table* → *Standardize*.

## Creating a Table Structure

### Prerequisites

First, create the characteristics you want to enter in the table in the characteristics maintenance functions.

The characteristics must be single-value, so that the table can be processed correctly in dependencies.

### Procedure

1. From the variant configuration menu, choose *Tools* → *Table structure* → *Create*.



You can also create a table structure by copying from an existing table structure.

- a) To do this, enter the name of the table you want to create and choose *Table* → *Copy from*.
- b) You see a dialog box in which you enter the name of the table structure you want to copy. You can change the copied data in the new table.
2. You see the initial screen. Enter a name for your table.  
Confirm your entries.
3. Maintain the basic data of the table.  
If you want the possible value combinations to be multiplied out when you display the table entries, select *Decision table* (see [Defining a Decision Table \[Page 188\]](#)).  
You can change this setting for your user when you maintain table entries.
4. Choose *Goto* → *Characteristics overview* to assign characteristics to the table.
5. Define the first value assignment alternative.  
Select the characteristics you want to use as key fields.
6. To enter more value assignment alternatives, choose *Goto* → *Val. assignment alt.* All value assignment alternatives except the first are only relevant to constraints. Actions and procedures always use the first alternative.
7. Save the table structure.

---

Changing a Table Structure

## Changing a Table Structure

### Use

You can insert new characteristics when you change a table structure. You can delete existing characteristics from the table structure, provided that they have not yet been used in dependencies.

You can only change value assignment alternatives if the table has not yet been used in released dependencies:

- You cannot change the first value assignment alternative once the table has been used in an action or procedure.
- If the table is used in constraints, you can no longer change any existing value assignment alternatives. However, you can create new value assignment alternatives.



You can change value assignment alternatives and delete characteristics if you set the status of the dependencies to *Locked*.

### Procedure

1. From the variant configuration menu, choose *Tools* → *Table structure* → *Change*.
2. Enter the name of the table whose structure you want to change.
3. Make your changes and save the table structure.

## Displaying a Table Structure

### Procedure

1. To display a table structure, choose *Tools → Table structure → Display*.
2. Enter the name of the table whose structure you want to display.
3. You can use the *Goto* menu to go to the individual maintenance screens.



In display mode, you can branch directly to the table maintenance function by choosing *Goto → Table maintenance*.

## Value Assignment Alternatives

## Value Assignment Alternatives

### Use

Value assignment alternatives are needed for setting characteristic values.

You define these when you set up your table structure. A value assignment alternative describes which table fields are key fields and which table fields are data fields. Key fields are used to find a table line whose data fields can be used to infer values.

In the following table, characteristic MODEL is defined as a key field to infer values for characteristics COLOR and LENGTH.

Characteristics	Value assignment alternatives
MODEL	X
Color	
Length	

### Features

#### Value Assignment Alternatives in Procedures (Actions)

You can only define one value assignment alternative for procedures (actions). Once the table has been used in a procedure (action) with *Released* status, you can no longer change the first value assignment alternative.

#### Value Assignment Alternatives in Constraints

##### Setting Values

In constraints, you can define several characteristics for which the constraint sets values. To set these values, value assignment alternatives must be maintained to allow correct access.



##### OBJECTS:

```
B is_a (300) Bike where MOD = Model; Typ = Bike_type; FH =
Frame_height.
```

##### RESTRICTIONS:

```
(MODEL = MOD, Bike_type = Typ, T_Frame_height = FH).
```

##### INFERENCES:

```
MOD, TYP, FH.
```

This constraint is for setting values for all three characteristics, so you need to define three value assignment alternatives:

Characteristics	A1	A2	A3
-----------------	----	----	----

Value Assignment Alternatives

MODEL	X		
BIKE_TYPE		X	
T_FRAME_HEIGHT			X

Once the table has been used in a constraint with '*Released*' status, you can no longer change existing value assignment alternatives. You can only create new value assignment alternatives for the table.



You cannot define a new value assignment alternative that is a subset of an existing value assignment alternative.

### Restricting Values

You do not need to define value assignment alternatives when you use tables in constraints to restrict allowed values.

### Unique Value

The key fields must set one value only, not two or more.

Table T1

MODEL	COLOR	LENGTH
<b>A</b>	<b>red</b>	<b>100</b>
<b>A</b>	<b>blue</b>	<b>100</b>
B	red	200
C	red	50

If the value assigned to MODEL is 'A', a single value cannot be inferred for COLOR. Characteristic COLOR can have either the value 'red' or the value 'blue'. For this reason, a value cannot be set.

**Multiple-Value Characteristics in Table Calls**

## Multiple-Value Characteristics in Table Calls

### Use

Only single-value characteristics should be used in tables. Multiple-value characteristics cause problems when dependencies are processed.

If a characteristic of the class is multiple value, create a single-value characteristic with the same data type and allowed values and reference it to the characteristic of the class.



Class BIKE contains multiple-value characteristic EXTRAS and single-value characteristic MODEL. You want to define a table so that the extras for the bike depend on the model.

To do this, you create characteristic EXTRAS\_SINGLE for the table. Characteristic MODEL is single-value, so it does not need a special characteristic for the table.

In the table, enter the value combinations for MODEL and EXTRAS\_SINGLE.

Create an action that compares the single-value characteristic with the multiple-value characteristic:

```
TABLE Extras  
(Model = Model,  
Extras_single = $Self.Extras)
```

### Features

#### Setting Values for a Multiple-Value Characteristic

A table must always set unique values. This means that you can only use a table to set one value for a multiple-value characteristic.

**Table Extras01**

MODEL	EXTRAS_SINGLE
JOURNEY	Stand
CITY	Basket
COUNTRY	Mudguard

The table cannot set both a stand and a basket as extras for one model.

#### Multiple-Value Characteristics in Conditions

You cannot use a condition to compare a single-value characteristic in a table with a multiple-value characteristic in a class, because this produces an ambiguous statement.

For example, you cannot use the following table call in a precondition:



## Multiple-Value Characteristics in Table Calls

```
TABLE Extras
(Model = Model,
Extras_single = Extras)
```

The system cannot interpret the comparison between EXTRAS\_SINGLE and EXTRAS, and therefore cannot tell whether the condition is fulfilled:

- If the value assigned to EXTRAS\_SINGLE is one of the values assigned to characteristic EXTRAS
- If the value assigned to EXTRAS\_SINGLE is the **only** value assigned to characteristic EXTRAS

### See also:

[Restricting Characteristics with a Variant Table \[Page 149\]](#)

## Link to a Database Table

# Link to a Database Table

## Use

You can link a variant table to a database table. The link between a variant table and a database provides new capabilities for accessing and processing data. System performance is improved considerably, especially when accessing individual points in large tables (for example, a dependency selects a specific table entry).

You can also buffer a database table, which can improve system performance even more.

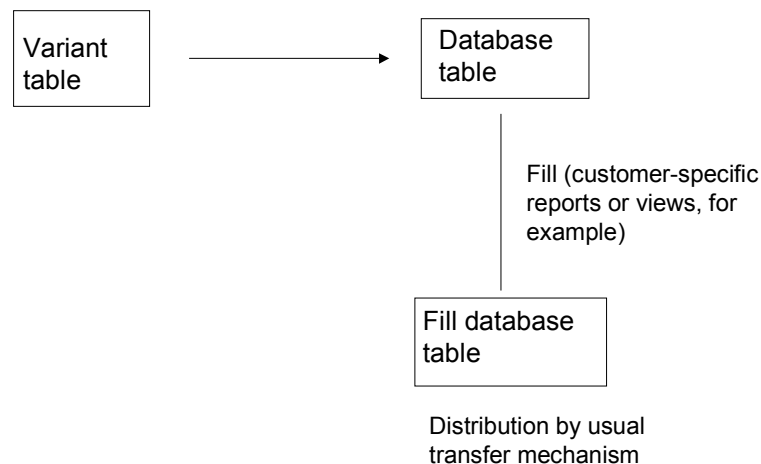
## Prerequisites

To link a variant table to a database table, you need authorization C\_LOVC\_DBI in your user master record. This authorization can be restricted to tables within a certain name range.

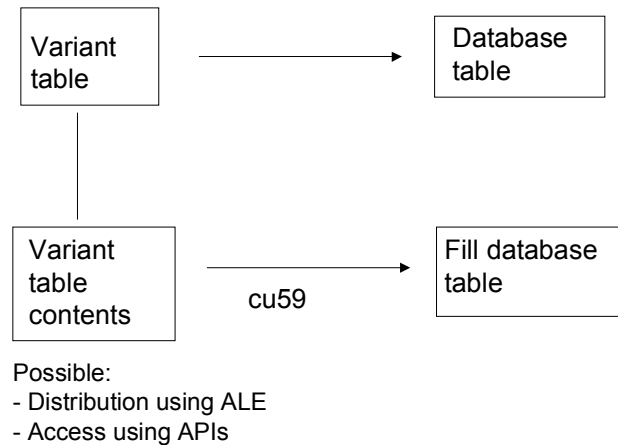
## Features

You have the following options:

1. Use an existing database table and create a variant table to link to this database table.



2. Copy an existing variant table to a database table and link the variant table to the database table.



If you create a database table:

- Once the link is activated, the contents of the old variant table are ignored. However, you can use this feature to prepare a new status of the table contents in table maintenance (CU60). The new status is then used later to fill the database table, and only becomes active in the live system at this point.
- If you define the client as a key field, the table contents are client-dependent. This can be useful, because variant tables are client-dependent.

This means that changes in the test client, for example, do not affect the live client.

- Decide which key fields you want the database table to have. Make sure that they match the variant table and its characteristics.

When defining database tables, make sure that each characteristic matches the database table field to which you assign it.

- The data types of fields must match the data types of the characteristics, but the system only checks that data types DATE and TIME are not assigned to each other incorrectly. The system makes no other checks. If the conversion does not work, the field value is automatically 0.
- Check the technical settings to see whether buffering the table will be useful in your environment.

You create the link when maintaining the table structure.

The database table fields must be assigned to the characteristics of the table.

**See also:**

---

**Link to a Database Table**

SAP Library *BC – ABAP Dictionary*, [Creating Tables \[Ext.\]](#)

## Linking a Variant Table to a Database Table

### Use

You link a variant table to a database table by maintaining the basic data of the table structure.

### Prerequisites

You have created a database table in the ABAP Dictionary.

To link a variant table to a database table, you need authorization C\_LOVC\_DBI in your user master record. This authorization must apply to the name range of the table.

### Procedure

1. In the basic data of the table structure, enter the name of the database table.

You see the *Link active* indicator.



You can only select this indicator once you have assigned characteristics to the table.

3. Select the indicator.
4. Choose *Goto* → *Field Assignments*, and assign the characteristics to the fields of the database table.
5. Save the table.



To log the user that changed the table and the date of the link to the database table, choose *Extras* → *Administrative data*.

### Result

You can use the table in dependencies. When the system accesses a table it goes directly to the database, so it can return a result more quickly.

If the database is new and does not contain any data yet, copy the contents of the variant table to the database table.

## Transferring Variant Table Contents to Database Table

## Transferring Variant Table Contents to Database Table

### Use

You can copy the contents of an existing variant table to a database table.

The following restrictions apply:

- You need authorization for this function, authorization for displaying tables, and authorization for maintaining table entries.
- The database table must be within the customer name range (beginning with a Y or a Z).
- Each key field in the database table (except the client) must have a corresponding characteristic in the variant table.
- The contents of the variant table must not be maintained with engineering change management.
- The variant table must not contain interval values.
- The characteristics must not be in exponential format 1 (standard) or 2 (Exponent entered).

### Prerequisites

The variant table must be linked to a database table. The link must be active.

### Procedure

1. From the variant configuration menu, choose *Tools* → *Table entries* → *Transfer to database table*.



This function executes report program RCCUVTDB.

2. Enter the database operation you want to execute:
  - *Insert* inserts new data records in the database.
  - *Insert and change* inserts new data records in the database and changes existing records.
  - Deleting data records is not currently supported.



Database Table

MATNR	WERKS	DFELD1
4711	0001	A

Data transfer:

1. Insert new data records: 4712/0002/B
2. Change existing data records: 4711/0001/B

---

Transferring Variant Table Contents to Database Table

## Maintaining Table Entries

# Maintaining Table Entries

When you maintain a table, you enter combinations of values for characteristics. You can maintain table entries with engineering change management. You can use change numbers with temporal validity (valid-from date).

## Prerequisites

You have created the table structure and assigned characteristics to the table.

## Procedure

1. From the variant configuration menu, choose *Tools* → *Maintain table*.
2. Choose *Maintain*.
3. Enter the name of the table.

If you want to use engineering change management to maintain the table entries, enter a change number. The change number must be active for object type *Variant table*.

Confirm your entries.

4. Enter valid value combinations for the characteristics you entered in the table structure.

To define how the table is displayed, choose *Table display* (see [Decision Table \[Page 188\]](#)).

5. Enter a separate table line for each possible combination of values.
  - Do not enter several values for one characteristic in one table field, because this entry cannot be interpreted.
  - Do not enter intervals. This can lead to problems in dependencies and adversely affect system performance.
  - To check that the table entries are entered correctly, choose *Table* → *Standardize*. If you have entered several values for a characteristic in one table line, this function multiplies out the value combinations, so that each line contains only one value for each characteristic.
6. Check the consistency of the table by choosing *Table* → *Check*.

The system checks for all value assignment alternatives whether unique values can be inferred from the key fields in the variant table. (see also [Value Assignment Alternatives \[Page 174\]](#)) The fields that are not defined as key fields in a value assignment alternative are ignored. If the system finds an error, you see an inconsistency message, and the table cells that are not unique are highlighted in color.
7. Save the table entries.

## Exporting Tables

You can display and process a table with EXCEL. You can also save the table as a PC file.

1. Choose *Table* → *Export*.
2. You see a dialog box in which can you specify:
  - Whether the table is saved to a file



## Maintaining Table Entries

- Whether the table is displayed in an EXCEL file  
Confirm.

The table is processed according to your specifications.

For more information on the technical requirements for exporting lists, see the section *BC List Export* in the SAP Library *ABAP Programming and Runtime Environment*.

## Changing Table Entries

## Changing Table Entries

### Procedure

To change table entries, choose *Tools → Maintain table* from the variant configuration menu.

1. You see the initial screen. Enter the name of the table. Choose *Table → Maintain*.  
If you want to make your change on a specific date, enter a valid change number.
2. You see a table with value combinations.
  - If you choose *Goto → Change overview*, you can see which values have been changed. If the changes were made with a change number, you see the change number and date.
  - You can define that only lines with specific values are displayed. To do this, choose *Extras → Filter*. You see a dialog box listing the allowed values of the characteristic. Select the characteristic values you require.
  - If you have use engineering change management and also defined that each line in the table is shown with the date when it was last changed, you see the date of the last change number used.
3. You can make the following changes to lines:
  - Insert
  - Duplicate
  - Delete

Select the line you want to change and choose *Edit → Entry*.

You can also create new entries.



If you use a change number, you can only change the lines that have not been changed with another change number that has the same valid-from date.

4. Save your changes.

## Displaying Table Entries

### Procedure

To display table entries, starting from the variant configuration menu, choose *Tools → Maintain table*.

1. You see the initial screen. Enter the name of the table you want to display.  
If you want to display the table on a specific date, enter the date you require.
2. To display the combinations of values for the characteristics, choose *Table → Display*.

## Decision Table

### Decision Table

Decision tables are a special display variant of tables. Before you can display the value combinations in the form of decision tables, you need to fulfill the following requirements:

- A finite number of values must be allowed for the characteristics. If characteristics with the *Additional values allowed* indicator are in the table, you cannot create a decision table.
- Each line must contain a value for each characteristic.
- Only one value can be entered for single-value characteristics. You can enter several values for a multiple-value characteristic. There is an AND relationship between these values.

Decision tables multiply out all the combinations of the values entered. The combinations can be displayed in the form of a matrix or as a list. The combinations you have already entered are selected. However, you can also select other combinations or deselect selected combinations. If you switch from the decision table back to the original display mode, you see the changed combinations there.

You can restrict further the values to be displayed for the combinations in a decision table by choosing *Goto* → *Value set*.

To display the decision table as a matrix, proceed as follows:

1. Choose the *Table display* pushbutton and select the *Decision table* indicator. Enter *Matrix* as the display format.  
Confirm.
2. You see all the possible combinations of the values entered as a matrix.  
You can select the valid combinations in the bottom line. The selected columns are displayed in color.  
You can select a combination and display it in the first column.
3. You can change the display format of the decision table at any time and display the combinations as a list.  
To do this, choose the *Table display* pushbutton and select the *List combinations* indicator.  
You see the valid combinations as lines of a list. You can select valid combinations using the plus or minus sign at the end of each line.

## Tables in Actions and Procedures

### Purpose

In actions and procedures, tables are used to infer values.

### Prerequisites

- Actions and procedures only read the first value assignment alternative defined in the table structure.
- Values can only be inferred for a characteristic that refers to the data field of the first value assignment alternative.
- You must enter the variable \$SELF for characteristics of the class for which you want to infer values.
- You must ensure that exactly one value is inferred when you access a table (see [Value Assignment Alternatives \[Page 174\]](#)).

### Process Flow

The following example is for an action. The tables calls in actions are the same as the table calls in procedures.

Configurable material BIKE has characteristics SADDLE, SADDLE\_SUPPORT, and SUPPORT\_MATERIAL. The data for the saddle can determine the type of SADDLE\_SUPPORT.

1. Choose *Tools* → *Table structure* → *Create*.

Create table T\_SADDLE and assign your 3 characteristics to it.



All characteristics are single-value, so you use the same characteristics in the table and in the class.

Characteristic SADDLE is defined as a key field in the value assignment alternative:

Characteristic	Key Field
SADDLE	X
SADDLE_SUPPORT	
SUPPORT_MATERIAL	

2. Choose *Tools* → *Table entries* and enter the following combinations of values:

SADDLE	SADDLE_SUPPORT	SUPPORT_MATERIAL
SR_ErgoGel	Patent support	Steel
SR_Gel	Standard support	Steel

**Tables in Actions and Procedures**

Touring_saddle	Standard support	Aluminum
MTB_saddle	Special patent support	Aluminum

3. Create action ACT\_SADDLE\_SUPPORT. The characteristics in the class are compared with the characteristics of the table. First, you refer to the characteristics in the table, then to the characteristics in the class. The characteristics of the class are referred to with \$SELF:

**TABLE T\_SADDLE**

```
(SADDLE = SADDLE ,  
SADDLE_SUPPORT = $SELF.SADDLE_SUPPORT ,  
SUPPORT_MATERIAL = $SELF.SUPPORT_MATERIAL)
```

4. Allocate this action to the configuration profile.

**Result**

In the class, assign a value to characteristic SADDLE. The table call in the dependency finds the correct table line and infers values for SADDLE\_SUPPORT and SUPPORT\_MATERIAL.

## Tables in Constraints

### Purpose

You can use tables in constraints to:

- Restrict the allowed values of characteristics using restrictable characteristics (see [Example: Restricting a Characteristic with a Table \[Page 150\]](#)).
- Infer values.
- Enter conditions in the CONDITIONS section.

### Inferring Values

You can use tables in constraints to set characteristic values, as in actions and procedures. The difference is that you can process the table in several different directions in constraints, because you can define different value assignment alternatives.

### Process flow

Configurable material BIKE has characteristics DYNAMO and HEADLAMP. The values of these 2 characteristics influence each other.

1. Choose *Tools* → *Table structure* → *Create*.

Create table T\_LIGHT and assign your 2 characteristics to it.



All characteristics are single-value, so you use the same characteristics in the table and in the class.

The difference is that you can define several value assignment alternatives, so you can use either characteristic to infer a value from the other.

Characteristics	Alternative 1	Alternative 2
HEADLAMP	X	
DYNAMO		X

2. Choose *Tools* → *Table entries* and enter the following combinations of values:

HEADLAMP	DYNAMO
Halogen	Elektra
Lumotec	Axa
FER	FER dynamo

3. Create a dependency net with the following constraint:

**Tables in Constraints**

OBJECTS:

BK IS\_A (300)BIKE

RESTRICTIONS:

TABLE T\_LIGHT

(HEADLAMP = BK.HEADLAMP,

DYNAMO = BK.DYNAMO)

INFERENCES:

BK.HEADLAMP, BK.DYNAMO

4. Allocate the dependency net to the configuration profile.

**Result**

As soon as you assign a value to characteristic HEADLAMP or DYNAMO, the value for the other characteristic is inferred by the table call.



## Tables in Preconditions

### Purpose

You can use table calls in preconditions to perform the following functions:

- Check the consistency
- Display possible entries



Value assignment alternatives are not relevant to preconditions, because preconditions are not used to infer values.

If you call a table in a precondition, the system only shows the values that are allowed according to the value combinations in the table.

### Process Flow

1. Choose *Tools* → *Table structure* → *Create* and create table EXTRAS. Assign characteristics MODEL and EXTRAS to the table.  
The table is for a precondition, so do not enter any value assignment alternatives.
2. Choose *Tools* → *Table entries* and enter the following combinations of values:

Table EXTRAS

MODEL	Extras
JOURNEY	Stand
CITY	Basket
COUNTRY	Mudguard

3. Enter preconditions for the values of characteristic EXTRAS:

PRE\_STAND

**Table Extras**

**(Model = Model, Extras = 'Stand')**

PRE\_BASKET

**Table Extras**

**(Model = Model, Extras = 'Basket')**

PRE\_MUDGUARD

**Table Extras**

**(Model = Model, Extras = 'Mudguard')**

4. Assign this precondition to characteristic EXTRAS. The values for EXTRAS are now only displayed for the appropriate models.

---

**Tables in Preconditions**

If you create one precondition instead of 3, and do not enter the 3 values explicitly, the system does not check whether the value is correct until you press ENTER during value assignment.

**Table Extras**

`(Model = Model, Extras = Extras)`

## Tables in Selection Conditions

### Purpose

You can use tables in selection conditions to define the combinations of values for selecting specific BOM items.

You have the following options for using table calls in selection conditions for BOM items:

- You can define a separate table with allowed value combinations for each BOM item. In the selection condition, you enter the table for the BOM item.
- You can define one table for all the BOM items. If you do this, you must enter the BOM item referred to in each line of the table. To do this, define a separate characteristic to include in the table.



Value assignment alternatives are not relevant to selection conditions, because selection conditions are not used to infer values.

### Prerequisites

You have created an additional characteristic that identifies the BOM item. The characteristic need not be entered in the classes involved.

### Process Flow

Configurable material BIKE has characteristics SADDLE\_SUPPORT and SUPPORT\_MATERIAL. These characteristics are required to select saddle supports S1001, S1002, S1003, and S1004.

To identify the BOM items in the table, you have created single-value characteristic COUNTER especially for the table.

1. Choose *Tools* → *Table structure* → *Create*.

Create table T\_SEL\_SADDLE\_SUPPORT and assign your 3 characteristics to it.



All characteristics are single-value, so you use the same characteristics in the table and in the class.

2. Choose *Tools* → *Table entries* and enter the following combinations of values:

COUNTER	SADDLE_SUPPORT	SUPPORT_MATERIAL
10	Patent support	Steel
20	Standard support	Steel
30	Standard support	Aluminum
40	Special patent support	Aluminum

**Tables in Selection Conditions**

3. You create 4 selection conditions for the individual materials:

SEL\_ITEM10

TABLE T\_SEL\_SADDLE\_SUPPORT

(COUNTER = '10', SADDLE\_SUPPORT = SADDLE\_SUPPORT,  
SUPPORT\_MATERIAL = SUPPORT\_MATERIAL)

SEL\_ITEM20

TABLE T\_SEL\_SADDLE\_SUPPORT

(COUNTER = '20', SADDLE\_SUPPORT = SADDLE\_SUPPORT,  
SUPPORT\_MATERIAL = SUPPORT\_MATERIAL)

SEL\_ITEM30

TABLE T\_SEL\_SADDLE\_SUPPORT

(COUNTER = '30', SADDLE\_SUPPORT = SADDLE\_SUPPORT,  
SUPPORT\_MATERIAL = SUPPORT\_MATERIAL)

SEL\_ITEM40

TABLE T\_SEL\_SADDLE\_SUPPORT

(COUNTER = '40', SADDLE\_SUPPORT = SADDLE\_SUPPORT,  
SUPPORT\_MATERIAL = SUPPORT\_MATERIAL)

4. Assign the selection conditions to the individual BOM items.

**Result**

A BOM item is selected as soon as the values assigned to the characteristics in the table lines matches the values for a counter in the selection condition.

## Creating Table Lists

### Use

You can produce a list of tables using the following selection criteria:

- Table name
- Table description
- Table status
- Table group
- Database table
- Created by (user)
- Last changed by (user)

### Procedure

To produce a table list, proceed as follows:

1. From the variant configuration menu, choose *Tools* → *Table structure* → *List*.

Enter the criteria according to which you want to produce your list.

2. Execute the function. You see a list of tables that match your selection criteria.

You also see whether the table is linked to a database table and whether this link is active.

- From the result screen, you can select a table and process the structure. You can use any of the following functions: *Display*, *Change*, and *Create*.
- You can also display the table entries.
- You can display a where-used list for a specific table.

## User-Defined Functions

# User-Defined Functions

## Purpose

Functions enable you to use your own function modules for checking values and inferring characteristic values.

In a function, you refer to an ABAP function module. The function module is used to copy the characteristics and characteristic values as table contents. In the function module, you can access all the usual options for further processing.



When you call a user-defined function module, SAP Variant Configuration no longer has control of possible error situations: the person who writes the function module can use all ABAP language elements, but has sole responsibility for the code.

You may want to create function modules for the following, for example:

- Complex calculations based on characteristic values in configuration
- Complex validity checks for allowed values



You can use a function to concatenate values for characteristics to form a text string. The value for characteristic **Door\_ID** can be defined as a string formed from a concatenation of the values for characteristics **Door\_Material**, **Door\_Width**, and **Door\_Height**.

In a function module, you can define that the values for characteristics **Door\_Material**, **Door\_Width**, and **Door\_Height** are concatenated to form the value of characteristic **Door\_ID**. To do this, you define a function module with the following call:

'concatenate Door\_Material Door\_Width Door\_Height into Door\_ID'

You then create the function, and define this function module in the function. You include characteristics **Door\_Material**, **Door\_Width**, **Door\_Height**, and **Door\_ID** in the function and define the value assignment alternatives. In this case, you must define **Door\_ID** as a data field, because values for this characteristic are to be inferred.

## Integration

You can use functions in selection conditions, preconditions, actions, procedures, and constraints.

- In selection conditions, preconditions, constraints, and the conditional part of actions and procedures, functions are used to check the consistency of the values entered.
- In actions, procedures, and constraints, you can use tables to infer values.

## Features

There are 2 steps to creating functions:

## User-Defined Functions

1. You define a function module to process characteristics that are transferred across an interface.
2. You create a function, entering the function module and the characteristics. If the function is for inferring values, define value assignment alternatives.  
  
The characteristics of the function can be the same as the characteristics of the class, if the characteristics are single value. If the characteristics of the class are multiple value, create single-value characteristics especially for the function.
3. Enter the function in object dependencies. The characteristics of the function are compared with the characteristics of the variant class.

```
FUNCTION <function name>
  (<characteristic of function> = <characteristic of class>,
   <characteristic of function> = <characteristic of class>)
```

## Call in a Precondition or Selection Condition

All the characteristics entered are interpreted as input parameters for the function module. The function module serves only to determine a yes/no result for these input parameters. If the condition is not fulfilled, the function module must signal this by using the predefined exception FAIL. It is not possible to infer values in this context.

## Call in an Action or Procedure

In an action or procedure, you can use a function call to infer values. In the value assignment alternative, specify which characteristics are import parameters and which are export parameters of the function module. Output parameters (return values) must refer to single-value or multiple-value characteristics of the class. If one of the input parameters does not have a value assigned in configuration, the function is not called.

Function calls in actions and procedures read only the first value assignment alternative, as do table calls.

## Function Call in Constraints

You can use a function call in the conditional part of a constraint (test for exception FAIL) and in the restrictions part (test and inference).

## Restrictions

- Functions for use in constraints or actions must not have any side effects on configuration. For example, input and output must be transported via the function interface.
- The characteristics in a function that you refer to in a dependency must always be **single value** – you can only refer to single-value characteristics in the interface.

## Creating a Function

# Creating a Function

## Prerequisites

You have created a function module that processes the input parameters.

## Procedure

1. From the variant configuration menu, choose *Tools → Function → Create*.
2. You see the initial screen.

Enter the name of the function module as the name of the function. The system uses the name to automatically link the function to the function module.

Confirm your entries.
3. Maintain the basic data of the function.

In the basis data, the relevant data of the function module is displayed. You see the name of the ABAP include with the source code.

  - The status of the function module must be active before you can use the function in dependencies.
  - You can go directly to the function for maintaining functions by choosing *Environment → Maintain funct. mod.*
4. To see the characteristics screen, choose *Goto → Characteristics*. On this screen, you can assign characteristics to the function. On this screen you also enter the first value assignment alternative:
  - Select the characteristics you want to use as input parameters for the function.
  - Do not select the characteristics that are output parameters.
5. To enter more value assignment alternatives, choose *Goto → Val. assignment alt.* All value assignment alternatives except the first are only relevant to constraints. Actions and procedures always access the first alternative (see [Value Assignment Alternatives \[Page 174\]](#)).
6. Save your function.



When you maintain a variant function, an exclusive lock is transmitted.

From the point of view of the dependency, ensure that the function used does not essentially change from the time of the syntax check until writing to the database.

A variant function with status *Locked* cannot be added to a dependency. This is because the syntax check in the dependency checks the status.

It does not check at configuration runtime whether the function is locked. The result of processing a dependency only depends on the status of the dependency itself, not on the status of elements used in it.



## Changing a Function

You can change a function with or without a change number. If you change a function with a change number, the change number only applies to the allocation of characteristics to the function. Changes to the basic data or the value assignment alternatives are not time-dependent.

You can only change value assignment alternatives if the function has not yet been used in released dependencies.

- You cannot change the first value assignment alternative once the function has been used in a released action or procedure.
- You cannot change any of the value assignment alternatives once the function has been used in released constraints. However, you can create new value assignment alternatives.

To change a function, proceed as follows:

1. From the variant configuration menu, choose *Tools* → *Function* → *Change*.
2. Enter the name of the function you want to change.

If you want to change characteristic allocations on a specific date, enter a change number.



Please note that the change number must support characteristics maintenance if you want to overwrite characteristics.

To see a list of change numbers that have been used to process the function on specific dates, choose *Goto* → *Change numbers*. You can select a change number from this list and copy it to your application.

3. You can display a list of dependencies in which the function has been used from the *Basic data* screen. To do this, choose *Environment* → *Where-used list*. You can select a dependency from this list and display a where-used list for the dependency.

On the *Characteristics* screen, you can also use the *Environment* → *Where-used list* to display a list of dependencies where individual characteristics are used. The *Environment* menu also provides the following reporting functions:

- a) Choose *Environment* → *Char. history* to see whether the characteristics have already been processed using engineering change management, whether a characteristic has been deleted using engineering change management, and whether the characteristic was overwritten for the function.
- b) Choose *Environment* → *Change documents* to see details of when the function was changed. This includes changes that were not made with engineering change management.

---

**Displaying a Function**

## Displaying a Function

To display a function, choose *Tools* → *Function* → *Display*.

1. Enter the name of the function you want to display.  
If you want to display the function on a specific date, enter a date. Otherwise, the function is displayed on today's date.
2. You can use the *Goto* menu to go to the individual maintenance screens.

## Interface of the Function Module

In a function, you refer to an ABAP function module whose input and output data is copied as table entries.

For information on how to create function modules, see the SAP Library *BC Basis* → *ABAP Workbench* → *ABAP Workbench: Tools* → *Function Builder*.



If you want to specify a function module in a function, the name of your function module cannot have more than 18 characters.

The ABAP interface of an external function module is hard coded and comprises the following elements:

### Interface Import Parameters

Parameter Name	Typing	Reference Field	Content
GLOBALS	LIKE	CUOV_00	Global parameters for calling a function. However, the list of fields currently only contains the date.

### Table Interface

Tables MATCH and QUERY are required to transfer characteristics to ABAP code and back.

Parameter Name	Typing	Reference Type	Content
QUERY	LIKE	CUOV_01	Table of input parameters and the expected output parameters
MATCH	LIKE	CUOV_01	Table of output parameters All partial fields except ATCIO must be filled in a MATCH entry (especially format ATFOR)

Structure **CUOV\_01** comprises the following fields:

- VARNAM (characteristic name)
- ATFOR (format of the value)
- ATWRT (alphanumeric characteristic value in internal format)
- ATFLV (numeric characteristic value)
- ATCIO (Indicator: input (I) or output (O) parameter)

The fields ATFOR, ATWRT, and ATFLV only have values assigned for input parameters.

### Exceptions:

**Interface of the Function Module**

FAIL	This exception shows that the condition represented by the function is not fulfilled.
INTERNAL_ERROR	This exception shows that a runtime error has occurred processing the function.



Do not enter any other import parameters or table parameters unless you define them as optional.

The type of dependency in which the function is used determines which data is transferred to the function module and which is returned:

- If you want to use a function with your function module for checking the values entered (for example, in preconditions and selection conditions, and in the conditional part of actions and procedures), enter the input values you want to check in structure QUERY. The function module then informs you whether the values are allowed or not (exception FAIL).
- If you want to use the function module for inferring values, enter the input values and the characteristics required in structure QUERY. The output values are returned in structure MATCH.



If you transport the product model into another R/3 System, you must maintain the module manually.

The following help functions are supported for accessing import parameters:

- CUOV\_GET\_FUNCTION\_ARGUMENT: Read characteristics
- CUOV\_GET\_FUNCTION\_ARGUMENT: Transfer characteristics

If you use these modules for modeling, you can access entries in table QUERY directly by name or enter a return value in table MATCH.

## Function Call

This example uses an action to show how you can use functions to infer values.

For example, you can use a function to link the values of several characteristics together, so that these values form the value of another characteristic.

In our example, a computer manufacturer wants to use labels on PCs to show at a glance which casing the PC has, how big the hard disk is, and which processor is installed.

The PC has characteristic CASING for the casing, CPU for the processor, and HD for the hard disk. The text for the label is the value of characteristic LABEL\_ID.

To calculate the value of characteristic LABEL\_ID, you create function module Z\_LABEL\_ID, and a function with the same name, and assign characteristics CASING, CPU, HD, and LABEL\_ID to the function. You define the first value assignment alternative such that characteristics CASING, CPU, and HD are input parameters for determining the value of characteristic LABEL\_ID.

You use this function in an action as follows:

```
FUNCTION Z_LABEL_ID
  (CASING = $ROOT.CASING,
   CPU = $ROOT.CPU,
   HD = $ROOT.HD,
   LABEL_ID = $SELF.LABEL_ID)
```

The characteristics on the left-hand side are characteristics of the function. The characteristics on the right-hand side are characteristics of the PC. Characteristic LABEL\_ID must be referred to with the variable \$SELF, because LABEL\_ID is a characteristic of the object currently being processed. Otherwise, values cannot be inferred. The default object \$ROOT is assumed for the other characteristics.

You allocate this action to the configuration profile of material PC.

As soon as values are assigned to characteristics CASING, CPU, and HD the function sets a value for characteristic LABEL\_ID.

You configure the PC, assigning the following values to characteristics CASING, CPU, and HD:

```
CASING = 'TW' (character format)
CPU = '586' (character format)
HD = 1275 (numeric format)
```

These values are transferred to function module Z\_LABEL\_ID as import parameters in table QUERY. In function module Z\_LABEL\_ID, the characteristics and values are read from the interface table using function module CUOV\_GET\_FUNCTION\_ARGUMENT.

If you want to format the value of a numeric characteristic as an integer (1275) and not as a decimal figure (1,275.00), the value transferred by function module CUOV\_GET\_FUNCTION\_ARGUMENT must be converted to an integer. To do this, use the command 'move value\_num to value\_int'.

If you want to use the value in calculations, it may be better to use the decimal figure, rather than convert the figure to an integer.

Characteristic names must be entered in upper case letters in code. Lower case characters are not converted automatically.

**Function Call**

The ABAP command CONCATENATE links the characteristic values together to form a character string. Individual values are separated by hyphens. This character string is transferred to ID\_CHAR. Function module CUOV\_SET\_FUNCTION\_ARGUMENT returns the value for characteristic LABEL\_ID and transfers this value to table MATCH.

The value 'TW-586-1275' is set for characteristic LABEL\_ID in this configuration.

The code in the function module for the function is as follows:

**Example of Code for Function Module Z\_LABEL\_ID**

function Z\_LABEL\_ID.

```

*''-----
*''*''Local interface:
*''  IMPORTING
*''      VALUE(GLOBALS) LIKE CUOV_00 STRUCTURE CUOV_00
*''  TABLES
*''      QUERY STRUCTURE CUOV_01
*''      MATCH STRUCTURE CUOV_01
*''  EXCEPTIONS
*''      FAIL
*''      INTERNAL_ERROR
*''-----

data: id_char like cuov_01-atwrt,           for the result of the concatenation
      value1 like cuov_01-atwrt,           for the characteristic 'CASING'
      value2 like cuov_01-atwrt,           for the characteristic 'CPU'
      value3_num like cuov_01-atflv,       for the numeric characteristic 'HD'
      value3_int(4) type N
      dash(1) type c value '-'.

*..initialize table with export parameters.....*
refresh match.

*..get value of input characteristic CASING
call function 'CUOV_GET_FUNCTION_ARGUMENT'
  exporting
    argument    = 'CASING'
  importing

```

```

        sym_val      = value1
tables
        query       = query
exceptions
        arg_not_found = 01.

if sy-subrc <> 0.
    raise internal_error.
endif.

*..get value of input characteristic CPU
call function 'CUOV_GET_FUNCTION_ARGUMENT'
    exporting
        argument     = 'CPU'
    importing
        sym_val      = value2
tables
        query       = query
exceptions
        arg_not_found = 01.

if sy-subrc <> 0.
    raise internal_error.
endif.

*..get value of input characteristic HD
call function 'CUOV_GET_FUNCTION_ARGUMENT'
    exporting
        argument     = 'HD'
    importing
        num_val      = value3_num
tables
        query       = query
exceptions
        arg_not_found = 01.

if sy-subrc <> 0.

```

**Function Call**

```
    raise internal_error.  
endif.
```

```
*..all numeric characteristics would need to be  
*..converted to integer fields, because the CUOV_GET_FUNCTION_ARGUMENT  
*..delivers them as floats !  
    move value3_num to value3_int
```

```
*..do the concatenation
```

```
concatenate value1  
    dash  
    value2  
    dash  
    value3_int
```

```
into id_char.
```

```
*..add result to the table of output characteristics
```

```
call function 'CUOV_SET_FUNCTION_ARGUMENT'
```

```
    exporting
```

```
        argument      = 'LABEL_ID'
```

```
        vtype         = 'CHAR'
```

```
        sym_val       = id_char
```

```
    tables
```

```
        match         = match
```

```
    exceptions
```

```
        existing_value_replaced = 01.
```

```
endfunction.
```



## Functions for Accessing the Dynamic Database

### Use

Functions may need to read or change data in the current configuration, in addition to the characteristic values transferred in the function interface.

Function group CUPR provides a series of function modules for this type of access. You can use the function modules in this function group to determine and change data of objects \$SELF, \$PARENT, and \$ROOT in the configuration structure.

You call functions with these function modules by using keyword PFUNCTION. You can only use them in procedures, because reading or changing configuration data with function modules in this function group may have undesired effects when processing dependencies, which must be avoided in all declarative dependency types (such as constraints).

### Features

Function modules in function group CUPR let you read, set, or delete characteristic values for an instance.

The access modules in function group CUPR require an internal instance number (INSTANCE) to identify the relevant objects in the configuration. Do not hard-code the instance number in an external program, because the number is assigned dynamically at runtime for the objects included in the configuration.

When you call a user-defined function with keyword PFUNCTION, the system transfers the internal instance numbers for the following objects in the configuration to the function being called:

1. SELF: the instance currently being configured
2. PARENT: the object immediately above SELF in the decomposition hierarchy
3. ROOT: the initial object (root of the decomposition)

These three fields are part of structure GLOBALS in the generic function interface for user-defined functions.

They correspond to the keywords \$SELF, \$PARENT, and \$ROOT used in configuration.



You can display the function modules in function group CUPR in the ABAP Workbench Repository Browser. For details of the function modules, see the function module documentation.

---

Creating a Function List

## Creating a Function List

### Procedure

You can produce a list of functions using the following selection criteria:

- Function name
- Description
- Status
- Group
- Created by (user)
- Last changed by (user)

To produce a list of functions, proceed as follows:

1. From the variant configuration menu, choose *Tools* → *Function* → *List*.  
Enter the criteria according to which you want to produce your list.
2. Choose *Execute*. You see a list that matches your selection criteria.
  - On the result screen, you can select an individual function to process. You can process functions using the functions *Display*, *Change*, and *Create*.
  - From the result screen, you can go to the function module for the function.
  - You can produce a where-used list for an individual function.

## Variant Conditions

### Definition

You can use variant conditions to influence the price of a configurable material depending on the characteristic values assigned.

### Use

You can use variant conditions in Sales and Distribution and Purchasing to define surcharges and discounts for the basic price.

### Structure

Variant conditions consist of a variant key and an amount that is identified by the variant key.

## Variant Conditions in Purchasing

## Variant Conditions in Purchasing

### Use

In Purchasing, you can use variant conditions to define surcharges and discounts for configurable materials that are procured externally. The surcharges and discounts depend on the values assigned to the characteristics in the sales order or (for material variants) the material master.



If you order a bicycle frame in the color 'Silver metallic', the vendor adds a surcharge of \$20. You create a variant condition called SILVER. When you select a frame in silver metallic, the variant condition is added to the price automatically.

### Prerequisites

You have created a characteristic that refers to table MMCOM, field VKOND. The characteristic is assigned to the variant class for the configurable material.



An object characteristic with a reference to the table MMCOM field VKOND is multi-value because several variant conditions can be entered.

In Customizing for *Purchasing*, you have defined a new access sequence (0014) and 2 new condition types (VA00, which is quantity-dependent, and VA01, which is a percentage). These condition types are in schema RM0000. You have set the *Variant condition* indicator.

### Process Flow

1. You create a purchasing info record for the externally-procured, configurable material, in which you maintain the variant conditions.
2. You create a procedure where you enter the reference from the characteristic to the structure MMCOM and the variant key.



CHAR\_VARCOND = 'SILVER' if PAINT = 'Silvermetallic'

- You have created a characteristic, CHAR\_VARCOND, that refers to table MMCOM, field VKOND.
  - The variant condition SILVER with the amount \$ 20 is maintained in the purchasing info record.
3. You allocate the procedures to either the characteristic values that trigger the variant conditions or the configuration profile, depending on the condition defined in the procedures.
  4. You configure the material:
    - In the material master, as a material variant
    - In the sales order

**Variant Conditions in Purchasing**

5. In material requirements planning (MRP), a purchase requisition is created for the material. The purchase requisition contains the configuration of the material.
6. Once released, the purchase requisition is converted to a purchase order. The net price for the material is displayed in the item overview of the purchase order. When you display the conditions for an item, you see the value of the variant condition that is included in the net price.

**See also:**

[Creating a Reference Characteristic \[Ext.\]](#)

---

Maintaining Variant Conditions in the Info Record

## Maintaining Variant Conditions in the Info Record

### Prerequisites

The material for which the info record is to be created or changed is configurable.

### Procedure

1. Choose *Master data* → *Info record* → *Create* or *Change*.
2. Enter the vendor, the configurable material, the purchasing organization, and the plant.
3. Press `ENTER`.
4. Make the necessary specifications (see [Creating an Info Record for a Material With a Master Record \[Ext.\]](#)).
5. Choose *Extras* → *Variant conditions*.
6. Enter a variant key.
7. Select the variant key and press `ENTER`.

A box containing the condition types defined for the variant key appears.

8. Choose the desired condition type and specify the period during which the discounts and surcharges are to apply.
9. Enter the amount and save.

### Result

The next time the material is needed in a sales order (i.e. the configurable material is either reconfigured in such a way that a characteristic value is reached or it is already so configured), a requisition is generated for it and converted into a purchase order, the system applies the variant conditions when determining the purchase order price.

#### See also:

[Variant Conditions in Purchasing \[Page 212\]](#)

## Variant Conditions in Sales

### Use

In Sales and Distribution, you can use variant conditions to define surcharges and discounts for configurable materials, depending on the characteristic values you assign.



If a customer wants a car with a sunroof, the net price is increased by a surcharge of \$2000.

You create variant conditions with reference to a material, a distribution channel, and a sales organization.

### Prerequisites

You have created a characteristic that refers to table SDCOM, field VKOND. The characteristic is assigned to the variant class for the configurable material.



An object characteristic with a reference to the table SDCOM field VKOND is multi-value because several variant conditions can be entered.

### Process Flow

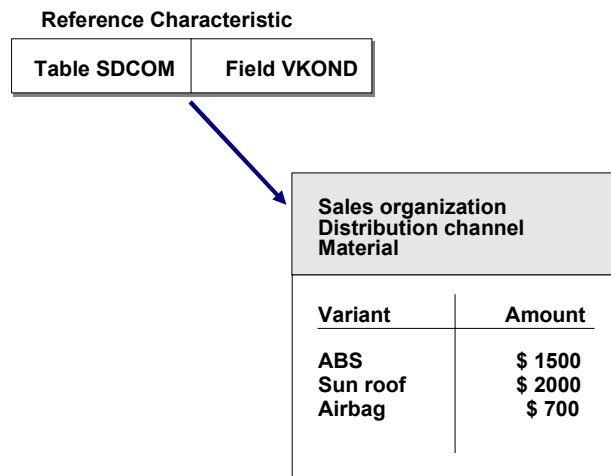
1. You maintain condition records for the variant conditions in the sales data of the material master record for the configurable material.
2. You create a procedure where you enter the reference from the characteristic to the structure SDCOM and the variant key.



CHAR\_VARCOND = 'ABS' if EXTRAS = 'ABS'

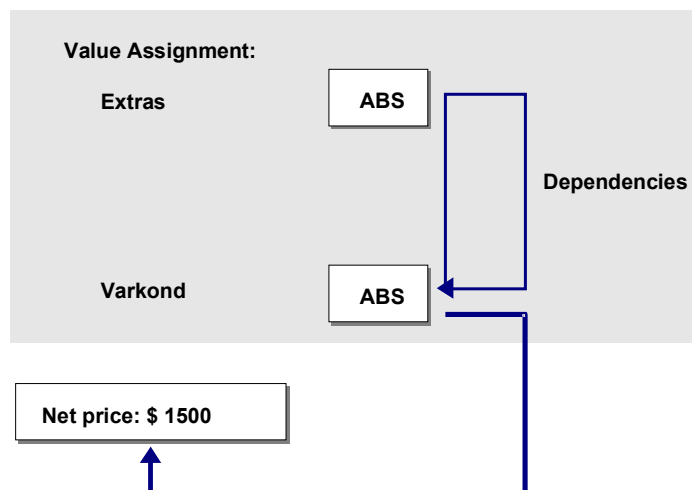
- You have created a characteristic, CHAR\_VARCOND, that refers to table SDCOM, field VCOND.
- The variant condition ABS with the amount \$1500 was maintained in the master data of sales and distribution.

## Variant Conditions in Sales



3. You assign the procedures to either the characteristic values that trigger the variant conditions or the configuration profile, depending on the condition defined in the procedures.
4. You configure the material in the sales order.
5. The net price for the material is displayed in the configuration editor.

If a value that triggers a variant condition is selected when you configure a material in a sales order, the price of the material displayed under *Net value* is automatically increased or reduced. In addition, the *Conditions* pushbutton is displayed. If you choose this pushbutton, you see which conditions have influenced the price.



See also:

[Creating Reference Characteristics \[Ext.\]](#)



## Maintaining Variant Conditions in Sales

### Use

You maintain a variant condition for a material with reference to a distribution channel and a sales organization. When you configure the material, the variant condition created for the material is accessed. Variant conditions are identified by a key.

### Procedure

1. To create a variant condition, choose *Logistics* → *Sales/distribution* → *Master data*, then *Conditions* → *Select by condition type* → *Create*.



You can also create variant conditions directly from the variant configuration menu by choosing *Environment* → *Pricing* → *Create conditions*.

2. Enter a condition type. The following condition types are defined for configurable materials:
  - VA00: this condition type expresses an absolute amount.
  - VA01: this condition type expresses the surcharge/discount as a percentage.

The prices are determined according to the placement of the condition category in the pricing procedure.

3. Confirm your entries. On the next screen, enter the variant key and the amounts to be identified by the key.

Enter the material, sales organization, and distribution channel for which you want to create the condition.

Enter the condition.



If you want a surcharge added if the value 'ABS' is assigned to characteristic EXTRAS, enter 'ABS' as the variant key and enter the amount by which the price is to increase.

4. Once you have entered all your variant conditions, save your entries.

For more information on conditions, see the R/3 Library *Pricing and Conditions*.

---

Assigning Variant Conditions Directly

## Assigning Variant Conditions Directly

### Prerequisites

If you want to assign exactly one variant condition to a characteristic value, you can assign the variant condition to the value directly. You need not create a procedure or action first. You create the variant condition in condition type VA00 or VA01.

You can only do this for characteristics with format CHAR. You cannot assign variant conditions to numeric characteristics directly.

### Procedure

1. In the variant configuration menu, choose *Environment* → *Configuration Simulation*.
2. Enter the configurable material and a plant, and choose *Configuration*, then *Chars*.
3. Select your characteristic.
4. Display the possible entries, and select the value to which you want to assign a variant condition.



The characteristic must not have an assigned value. If you have already assigned a value, close the possible entries and delete the value first.

5. Choose *Assign variant condition*.
6. You see a dialog box in which you enter the variant key.
7. Save your assignment.

In the configuration simulation, the variant key is shown for the value, not the amount.

### Result

In configuration in the sales order, the amount in the variant condition is displayed for the characteristic value.

If you select the value, the price is automatically increased by this amount.

## Variant Conditions in Procedures

If a bike is ordered with an adjustable frame, the price increases by 200.

Configurable material BIKE has characteristic EXTRAS with value '0015' (adjustable frame).

### Procedure

1. You have create variant 'FRAME' with the amount 200 in condition type VA00.
2. You have created characteristic VARCOND with a reference to structure SDCOM and field VKOND.
3. Create a single dependency of dependency type **procedure**.

In the procedure, you enter the characteristic with the table reference and the variant key as a value.

```
$SELF.VARCOND = 'FRAME'
```



Variant keys are case sensitive. In object dependencies, you must enter the variant key in exactly the same form as in the condition table. If the combination of upper and lower case letters does not match the table, the variant condition is not processed.

4. You allocate the procedure to characteristic value '0015' of characteristic EXTRAS.

### Result

If the value '0015' is selected for characteristic EXTRAS when you configure the bike, the procedure is processed. The procedure sets the value '0015' for characteristic VARCOND, and this value is used to add 200 to the net price.

## Variant Conditions with a Table

## Variant Conditions with a Table

The price of a bike increases according to the color. The colors are divided into different categories. Configurable material BIKE has characteristic COLOR.

1. Create characteristic SURCHARGE to refer to field VKOND in table SDCOM. The characteristic is multiple-value.
2. Create the following variant conditions as described in [Maintaining Variant Conditions \[Page 217\]](#):

Variant Key	Amount
CAT_A	200
CAT_B	300
CAT_C	400

3. Create a table and enter the combinations of colors and variant keys. Characteristic SURCHARGE is multiple value, so do not use it in the table (see [Multiple-Value Characteristics in Table Calls \[Page 176\]](#)).

Define additional characteristic T\_SURCHARGE. This characteristic has the same format as characteristic SURCHARGE. However, it is single-value and has no table reference.

4. Enter the following table entries:

For characteristic T\_SURCHARGE, enter the variant key for determining each color.



Values in a table are case-sensitive!

Table COLOR\_PRICE

COLOR	T_SURCHARGE
RED	CAT_A
BLUE	CAT_A
SCARLET	CAT_B
MATT_GREEN	CAT_B
FRENCH_LAVENDER	CAT_C

5. Create a procedure in which you call table COLOR\_PRICE.

You link the characteristics of the table to the characteristics of the class. Characteristic COLOR is the same in both.

**Table COLOR\_PRICE**

```
(T_SURCHARGE = $SELF.SURCHARGE,
COLOR = COLOR) .
```

6. Allocate the procedure to the configuration profile of the bike. Depending on which color you select, the price is increased by the amount maintained in the condition.



## Pricing Factors

# Pricing Factors

## Use

Surcharges or discounts can depend partly on a specific characteristic value, partly on other factors, such as length. This method of pricing can also be expressed in dependencies. To do this, you enter the factor by which you want the surcharge or discount to be increased or reduced, as well as entering the variant condition.



Pricing factors can only be maintained on characteristic level, not in the sales order.

The syntax of pricing factors is as follows:

```
$SET_PRICING_FACTOR ($SELF, <characteristic>, <variant key>, <factor>)
```

This expression contains the following information:

- The characteristic that refers to structure SDCOM, in which variant conditions are defined.
- The variant key used to infer the condition for a characteristic value.
- The factor by which the surcharge increases. You can enter the factor as a constant, a numeric characteristic, or a numeric expression.

## Features

You use built-in function `$SET_PRICING_FACTOR` in a procedure.

If you call `$SET_PRICING_FACTOR`, but the relevant condition is not set in the configuration, the factor is still processed. A dynamic database trace message tells you that the condition is missing. The missing condition is **not** set.

### Example:

1. The variant class contains characteristics WOOD and LENGTH (in meters).
2. You create characteristic VARCOND that refers to structure SDCOM and field VKOND, and assign this characteristic to the variant class.
3. You create variant condition OAK with amount 5 (currency of your choice) per meter.
4. You define a procedure so that variant condition OAK is set if the WOOD is 'Oak', and this variant condition is multiplied by the length times 3.

```
$SELF, VARCOND = 'OAK', IF WOOD = 'OAK'
```

```
$SET_PRICING_FACTOR ($SELF, VARCOND, 'OAK', LENGTH*3)
```

5. Allocate this procedure to the configuration profile of the configurable material.
6. If value 5 is assigned to LENGTH when configuring the material, variant condition OAK is multiplied by this factor ( $5 * 15 = 75$ ) and the value is added to basic price PR00.

## Dependency Group for Pricing

### Use

You can group together procedures that are relevant to pricing by using dependency group SAP\_PRICNG. These procedures are then only read by pricing functions. This improves system response times.

### Prerequisites

- You have created dependency group SAP\_PRICNG in Customizing for *Variant Configuration*.
- You assign procedures that are relevant to pricing to this dependency group.
- All procedures in this dependency group must be assigned to the configuration profile.

### Features

If your configuration model contains a large number of dependencies that are relevant to pricing, it is useful to assign them to this dependency group.

It is also useful to select the setting for pricing on request on the characteristic value assignment screen (see [Defining Settings for Pricing \[Page 57\]](#)).

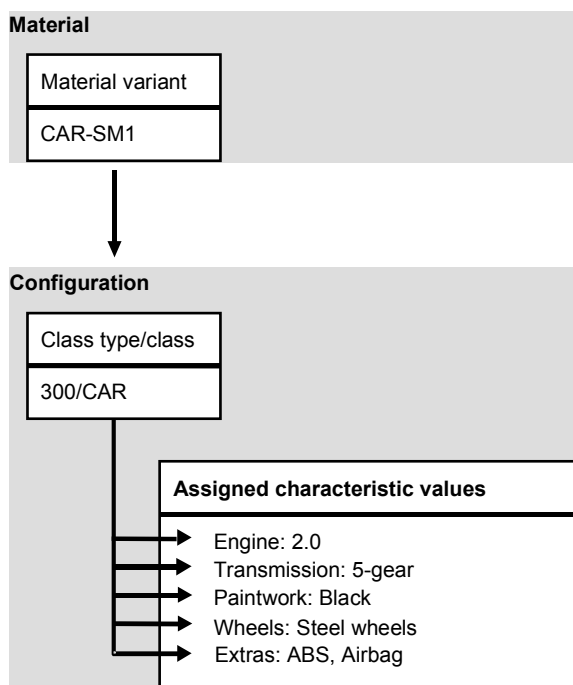
## Material Variants

## Material Variants

### Definition

A material variant is a material that can be kept in stock and that arises from an individual configuration of a configurable product.

The material master record of a material variant is linked to the configurable material and configured using the characteristics of the configurable material. This variant of the configurable material can then be manufactured and kept in stock.



### Use

If specific configurations or of a configurable product occur frequently, it is useful to create individual stock materials for these configurations, so that these materials can be manufactured in advance, and a sales order with this configuration can be supplied from stock if required.

In the sales order, you can do type matching for variants, and replace configurable materials with material variants (see [Variant Matching in the Sales Order \[Page 273\]](#)).

On the value assignment screen, you can check whether the values assigned match the configuration of a material variant (see [Settings for Variant Matching \[Page 60\]](#)).

To allow you to change the configuration of material variants in the sales order, the *Configuration allowed or required* indicator must be set in Customizing for *Sales and Distribution*. The requirements class is determined using the strategy group and requirements type defined in the material master.



## Structure

For material variants, a separate material master record is created with a material type that is kept in stock.

You can create a separate BOM and routing for a material variant, or you can link the material variant to the BOM and routing of the configurable material. The correct BOM items and operations are determined from the characteristic values assigned to the variant.

---

Maintaining Material Master Records for Variants

## Maintaining Material Master Records for Variants

### Prerequisites

A configurable material exists, for which you create a material variant. The configurable material has all the necessary settings in the configuration profile, class assignment, dependencies, and so on.

### Procedure

1. From the material master menu, choose *Material* → *Create general* → *Immediately*.

Enter a material number, an industry sector, and a material type for materials kept in stock.

You must process the following views:

- Basic Data
- Sales Data
- MRP Data



For material variants, you maintain the usual material master data, such as availability check, strategy groups, and so on. This documentation describes the data that you need to maintain specifically for material variants.

2. Maintain the required basic data.

Do **not** select *Material is configurable* for a material variant.

In the basic data, you can link the material variant to a configurable material. This configuration applies to all plants. To manufacture the material variant in-house, you must maintain the variant at plant level, so that BOMs, routings, and so on can be determined. Cross-plant variants can be used in Purchasing, but are not supported in Sales.

3. Maintain the sales data.



You can only use the variant in sales orders if you maintain sales data for the variant.

4. In the MRP data, enter a configurable material for configuring the variant at plant level.
5. Choose *Configure variant*.

You see the value assignment screen with the characteristics of the configurable material.

Configure the variant.

6. Once you have maintained all the data, save your material variant.

#### See also:

SAP Library *PP Demand Management* [Strategies for Variants \[Ext.\]](#)



---

**Bill of Material (BOM)**

## Bill of Material (BOM)

To do a requirements explosion for a configured material, you must maintain the data in the material master record and link a BOM to the configured material.



If the material of the material variant is the same as the configurable material, the link is already there.

You can either create a BOM especially for the configured material (perhaps using the copy from function) or allocate the configured material to the BOM for the configurable product. The second option is the more common solution.

If you allocate the configured material to the BOM for the configurable product, the BOM for the configurable product is interpreted and exploded according to the characteristic values assigned. The interpretation includes the processing of selection conditions and the setting of field values in BOM items according to the relevant actions.

### Allocating a Configured Material to a BOM

To allocate the configured material to the BOM of the configurable product, proceed as follows:

1. From the bills of material menu, choose *Bills of material* → *Material BOM* → *Alloc. config. mat.* → *Create*.
2. Enter the name of your configured material in the *Material* field.  
You see a screen on which your configured material is shown as the first allocation. Select the allocation and choose the *All allocs for BOM* function to see all the materials allocated to this BOM.
3. Save your allocation.

## Task List

To produce a configured material, you need to allocate a task list to the configurable material once you have created the material master.



If the material of the material variant is the same as the configurable material, the link is already there.

You can either create a task list especially for the configured material (perhaps using the copy from function) or allocate the configured material to the task list for the configurable product in the task list maintenance function. The second option is the more common solution. If you allocate the configured material to the task list for the configurable product, the task list for the configurable product is interpreted and exploded in the production order according to the characteristic values assigned. The interpretation includes the processing of selection conditions in the operations or activities, sub-operations, sequences of operations, and production resources/tools (PRTs).

### Allocating a Task List to a Configured Material

To allocate the configured material to the task list of the configurable material, proceed as follows:

1. From the routings menu, choose *Routings → Routing → Change*.
2. Enter the number of the configurable material.
3. Choose *Goto → Overview*.
4. Choose *Routing → Material allocation → New entries* and enter the number of the configured material you want to allocate to the task list.
5. Save your allocation.

## Creating Cross-Plant Material Variants

## Creating Cross-Plant Material Variants

### Use

For material variants that are procured externally, you can define the assignment to a configurable material in the basic data of the material master, because no plant-specific data needs to be determined for these variants.

### Procedure

1. In the material master menu, choose *Material* → *Create (General)* → *Immediately*.  
Enter a material number, an industry sector, and a material type for materials kept in stock.  
Select the views that you want to process.
2. Maintain the required basic data.  
Do **not** select *Material is configurable* for a material variant.
3. Under *Client-specific configuration*, enter the configurable material of which you are creating a variant.
4. Choose *Configure variant*.  
You see the value assignment screen with the characteristics of the configurable material.  
Configure the variant.
5. Once you have maintained all the data, save your material variant.

### Result

The variant can be entered in a purchase order. The configuration of the variant is copied from the material master.



Cross-plant material variants can be used in Purchasing. However, they are not supported in Sales.

## The Configuration Simulation

### Purpose

You can use the configuration simulation to check your configuration model. In the configuration simulation, you can test whether you have created the objects correctly and whether your dependencies work.

The BOM for the configuration result is selected according to the BOM application in the configuration profile.

### Simulation of Sales/Engineering

The configuration parameters in the configuration profile apply. However, you can define whether the configuration is simulated from the sales point of view or the engineering point of view:

- If you select *Sales & distribution*, you simulate the configuration in a sales document (sales order or quotation), so the BOM explosion depends on the settings in the configuration profile (see [Configurable Materials in Sales Documents \[Page 262\]](#)).
- If you select *Engineering*, you simulate the configuration in order BOM processing. The configuration and BOM explosion depend on the configuration parameters in the same way as when you maintain an order BOM. For example, you cannot configure the header material, because the configuration is copied from the sales order.

### Simulation of a Planned Order

On the characteristic value assignment screen, you can choose *Plnd order* to display the components that are determined in material requirements planning (MRP) according to the characteristic values assigned.

This is especially relevant to assemblies whose BOMs are exploded in MRP, not in *Sales & distribution* (for example, assemblies with the BOM explosion setting *None* in their configuration profile).

### Features

- To improve system performance when exploding a BOM on the result screen, you can restrict the level of detail.
- You can copy existing configurations from material variants, sales orders, or production orders.
- You can define whether the BOM or task list is exploded on the configuration result screen.

### Integration

You can only use the configuration simulation for materials.

#### See also:

[The Characteristic Value Assignment Screen \[Page 239\]](#)

[The Configuration Result \[Page 257\]](#)

---

The Configuration Simulation



## Simulating BOM/Task List Explosion

### Use



In the configuration simulation, you can check the explosion of BOMs and task lists (such as routings). On the initial screen, you can define whether the BOM or task list is exploded.

If you select *BOM*, the entire BOM structure is exploded according to the settings in the configuration profile. You can display the individual assemblies on the result screen.

If you select *Task list*, only the task list of the header material is exploded, not the task lists of the assemblies.

### Procedure

1. On the initial screen of the configuration simulation, select *BOM* to display the entire structure on the result screen.

If you have created more than one configuration profile for a material, you see a dialog box. Select a profile and choose  *Continue*. If you want to call detailed information on the profile, choose  *Profile Detail* or double-click on the profile.

2. To display the task list of an assembly on the result screen, select the assembly and choose *View* → *Objects* → *Task list*.

You see the operations or activities in the task list that were selected for the assembly during configuration.



Operations that have been changed by dependencies have an information icon next to them. You can display changes by choosing *Information*.

3. To return to the BOM explosion, choose *View* → *Objects* → *BOM*.

---

**Selecting Configured Objects to Copy**

## Selecting Configured Objects to Copy

A configurable material may have been configured in sales orders, production orders, or material variants.

You can copy the characteristic values assigned to the configurable material in these objects to your simulation.

### Procedure

1. From the variant configuration menu, call *Environment → Configuration simulation*.  
If you have not yet entered a plant in the current session, you see a dialog box. Enter the plant for your configurable material and confirm your entry.  
Enter the material you want to configure.
2. On the initial screen, choose *Configured objects*.  
This call the material variants, sales orders, or production orders that have been configured for this material, so that you can copy them to your configuration.
3. Select the type of object you want to copy from and choose the appropriate pushbutton.  
You see the existing variants of that object type (for example, all the sales orders).
4. Select a variant.  
The object (for example, order number) is copied to the dialog box.
5. Confirm your entries in the dialog box.  
On the initial screen, you see the object you selected to copy from.

### Result

You see the configuration you copied on the value assignment screen. You can change the values.

## Level of Detail for the BOM

### Use

You can define how many levels of a multi-level BOM structure are exploded on the configuration result screen. For example, if you only want to see the top two levels, you can hide the others. This reduces the response time for displaying the result screen.

### Integration

This function is integrated in the configuration simulation and order BOM processing functions.

### Features

If you restrict the level of detail for a BOM structure, you can use the same function on the result screen to change the settings again.

A traffic light icon shows you whether the BOM explosion is restricted:

- If the BOM is not exploded in full, you see a red traffic light on the result screen.
- If the BOM is exploded in full, you see a green traffic light on the result screen.

To display a specific assembly of the BOM on the result screen, choose *Assembly*.

1. Select the assembly and choose *Assembly*.
2. The display jumps to the place in the BOM structure where this assembly is shown.

---

Simulation of Costing

## Simulation of Costing

### Use

You can use the configuration simulation functionality to determine the costs for manufacturing a specific variant of the configurable material. Please note that this configuration is only a simulation – the costing data for a variant is not saved.

You can use the function to simulate the costing for different characteristic values:

- You can simulate costing for the selection of different variable parts. If the material does not have any characteristics with the *Entry required* indicator, you can also simulate costing for the non-variable parts of the BOM.
- You can use the *Values → Configurator → Inactive* function to switch off the checks made when characteristic values are assigned by dependencies. This allows you to cost variants that are not currently supported, but that you may want to produce in future.

Costing covers all levels of the BOM for the header material. Components that are selected from a class item are also costed.

You see a dialog box in which you must enter a costing variant. The system uses the costing variant to determine and value the quantity structure (for example, BOM and routing) to be costed. Use a costing variant that is defined for sales order costing. You define costing variants in Customizing for *Product Cost Controlling*.

### Prerequisites

Before you can cost a product, the *Product costing* indicator must be set in the material master records of the configurable material and its components, in the costing data. The accounting data must also be maintained.

### Activities

To call costing from the value assignment screen, choose *Environment → Costing*.

#### See also:

R/3 Library *CO Product Cost Controlling*.

## Creating a Routing with the Configuration Simulation

### Use

If you display the result of a configuration simulation as a routing explosion, you can use the result to create a new routing.

### Features

The following data is copied from the routing explosion result to your new routing:

- Header
- Sequence
- Operations and their:
  - Sub-operations
  - Material components
  - Production resources/tools
  - Inspection characteristics

You can edit the new routing and save it in the usual way.

**See also:**

*PP Routings*

---

Creating a Routing from the Configuration Simulation

## Creating a Routing from the Configuration Simulation

### Prerequisites

You have created a super routing for the configurable material.

### Procedure

1. From the variant configuration menu, choose *Environment* → *Configuration simulation*.  
Enter the configurable material.
2. Choose *Characteristics*, to go to the configuration editor.  
Configure the material.
3. Choose *Result* to see the BOM explosion result.  
In the BOM, select the material whose routing you want to display.
4. Choose *View* → *Objects* → *Task list*.  
You see the operations that were selected for the material during the configuration process.
5. Choose *Environment* → *Result as task list*.  
You see the *Initial Screen: Create Routing from Configuration* dialog box.
6. Process this screen and choose *Continue*.  
You see the *Create Routing: Check Header* screen.
7. Process this screen and choose *Continue*.  
You see the *Create Routing: Operation Overview* screen.  
The system proposes the operations that were selected for the material during the configuration process as a default.
8. Process this screen and save your routing.  
You return to the configuration simulation.

### Result

You have created a new routing.

## The Characteristic Value Assignment Screen

### Definition

The characteristic value assignment screen shows the characteristics of a configurable object, so that you can assign values to them.

### Use

On the characteristic value assignment screen, you define the features of a variant of a configurable product. You configure the material by assigning values to its characteristics.

As you enter characteristic values, dependencies for the characteristics, characteristic values, and the configuration profile are processed. Depending on the values you assign, characteristics or characteristic values can be shown or hidden, or values can be set for characteristics automatically.

You can define user-specific settings for the value assignment screen (for example, display of characteristics and characteristic values). If you do not define user-specific settings, the object-specific settings from the configuration profile apply.

### Integration

The characteristic value assignment screen in variant configuration is called by the following R/3 application components:

- LO Variant Configuration, in the configuration simulation
- Material master, when you create a material variant
- SD Sales and Distribution, when you create a quotation or sales order for a configurable material
- PM Plant Maintenance, when you enter a configurable general maintenance task list in a maintenance order or service order
- PM technical systems structuring, when you create an equipment with reference to a configurable material
- PS Project System, when you create a network or subnetwork from a configurable standard network

## Configuring Objects

## Configuring Objects

### Procedure

1. On the value assignment screen, values may already be assigned to some characteristics. These values are defined in characteristics maintenance as default values for the characteristic.

Either accept these default values or enter another value for the characteristic.

The default value may already have triggered dependencies.

- The characteristic value assignment screen may not show all characteristics, because some are hidden in accordance with the settings defined in characteristics maintenance, or are not relevant to the organizational area.

To change the restriction to organizational areas, choose *View → Organizational areas*.

2. To assign values to the characteristics, enter values on the value assignment screen or display the possible entries and select one.
3. Confirm your entries.
4. All the dependencies are processed (refer to [Processing Sequence for Object Dependencies \[Page 242\]](#)).
  - In the possible entries, you see an icon next to the characteristics and values that have object dependencies. To see the dependencies, choose this icon.
  - Dependencies can cause characteristics and characteristic values to be hidden or shown.
  - Dependencies can set values.



You cannot manually overwrite values that were set by dependencies.

5. If a value is set that violates a dependency, you see the *Inconsistencies* pushbutton.
6. To see what caused the inconsistency, choose *Inconsistencies*.
7. The assigned values are logged. To undo the values step by step, choose *Undo*.

### Status Display

When you exit the configuration editor, you may see a prompt with characteristics that have no value, asking you to assign values. The system registers *Incomplete* status.

This may be for the following reasons:

1. A characteristic that is defined as *Required* in the Classification System (required characteristic) has no value assigned to it.
2. A required characteristic that is not displayed because of a precondition has no value assigned to it.
3. A characteristic that must have a value assigned to it because of a selection condition has no value.



## Configuring Objects

Assign values to these characteristics or set the status of the configuration to *Locked* – otherwise, you cannot exit the configuration editor. You can change the status by choosing the pushbutton for showing the current status, or by choosing *Extras* → *Change status*.

In a sales order, *Incomplete* or *Locked* status can affect material requirements planning (see [Transfer of Requirements for Locked Configurations \[Page 275\]](#))

**Example**

A car manufacturer makes the color of seat covers a required characteristic. However, the manufacturer overlooks that fact that the color does not need a value for material "leather", because the company only makes leather seat covers in black.

In the sales order, it is too late to test whether a certain combination causes inconsistency. You need to check whether the assigned values are complete. This is why the configuration simulation is important, because the check is started automatically when you exit the configuration editor.

## Processing Sequence for Dependencies

# Processing Sequence for Dependencies

## Use

Each time you enter a value for a characteristic on the value assignment screen and confirm the value, the system processes all the dependencies.

The dependencies are processed in the following sequence:

1. All actions several times, until no more values can be inferred.
2. All procedures exactly once in the following sequence:
  - a) Procedures for the configuration profile in the order you defined
  - b) Procedures for characteristics
  - c) Procedures for characteristic values



The procedures for each characteristic or value are processed in the order you defined.

3. Actions

Values set by procedures can also trigger actions. For this reason, all actions are processed again.

4. Preconditions
5. Selection conditions for characteristics

Constraints are processed in parallel with points 1–3, if all the objects in the OBJECTS section are present, and if all the conditions in the CONDITIONS section are fulfilled.

## Integration

- You can display how the dependencies were processed by choosing → *Extras Trace*.
- You can switch off processing of object dependencies on the result screen by choosing *Value assignment* → *Configurator*.

## Prerequisites

Dependencies must be allocated to the characteristics and characteristic values.

## Explanation Functions for Value Assignment

### Use

To see information on the values assigned to characteristics in the dependency editor, choose *Explanation*.

- If you choose *Explanation* for a characteristic with a value, you see the author of the value:
  - User
  - Action
  - Procedure
  - Constraint
- If you choose *Explanation* for a characteristic without a value, you see any constraints that infer values for the characteristic.

## Trace Function

# Trace Function

## Use

A trace function for configuration lets you trace the internal processing steps that the system makes when processing dependencies.

## Features

You can start the trace function for the following objects:

- Single dependencies
- Constraints
- Tables and functions
- Dynamic database

Additional filter criteria:

- You can restrict the trace function for single dependencies to specific dependencies.
- You can restrict the trace function for the dynamic database to a given list of characteristics.
- You can restrict the trace function for tables and functions to a given list of tables or functions.

You can define the level of detail for your trace.

## Activities

- You activate the trace function in the configuration editor by choosing *Extras → Trace → Activate*.



For configuration simulations, you can start the trace on the initial screen.

- After each assigned value, you can display the dependencies that were processed by choosing *Extras → Trace → Display*.

Since the dependencies are reprocessed each time a value is assigned, you see the same dependency several times.

## Configuration Buffer

When you configure an object, you can save the values currently assigned and call them up again if required. This allows you to store various in-between stages of a configuration.



As soon as you exit the configuration, the stored values are deleted.

## Procedure

1. In the configuration editor, choose *Values → Configuration buffer → Save as*.

You see a dialog box in which you save the configuration so far under a specific name.



*Values → Configuration buffer → Save* saves the configuration of an object under an internal name. You can only use this function to store one configuration.

2. To see all the configurations of the object for the current session, choose *Values → Configuration buffer → Overview*.
  - a) You can select a configuration of the object.
  - b) You can delete a configuration of the object.

## Interface Design – Overview

## Interface Design – Overview

### Use

This function lets you group characteristics together and define a characteristic value assignment screen for a group of characteristics with the settings you choose. First, you group together certain characteristics on the value assignment screen. Then you can create a special screen for these characteristics, and arrange the characteristics and values as you require.

### Prerequisites

- Your user master record contains authorization object C\_LOVC\_DSG with the activity that you want to perform.
- The configuration profile contains a name for an interface design under *Config initial screen* → *UserInterf*. This activates the function on the value assignment screen.

### Features

The following list shows the different options for grouping characteristics together and arranging them for an interface design.

Interface design	Display option	Summarize	Design	Restrictions
Characteristics group	List	Yes	<ul style="list-style-type: none"> <li>- Move characteristics and values</li> <li>- Define border</li> <li>- Screen size of list</li> </ul>	<ul style="list-style-type: none"> <li>- No automatic summarization of characteristics, if dependencies hide characteristic</li> <li>- Only the first values is displayed for multiple-value characteristics</li> </ul>

Pushbutton	List	Yes	<ul style="list-style-type: none"> <li>- Move characteristics and values</li> <li>- Define border</li> <li>- Screen size of list</li> </ul>	<ul style="list-style-type: none"> <li>- No automatic summarization of characteristics, if dependencies hide characteristic</li> <li>- Only the first values is displayed for multiple-value characteristics</li> </ul>
Tab	Tab	Yes	<ul style="list-style-type: none"> <li>- Only grouping of characteristics</li> </ul>	<ul style="list-style-type: none"> <li>- No design possible</li> <li>- Only grouping and sequence within tab</li> </ul>
Sequence	-	No	<ul style="list-style-type: none"> <li>- Normal value assignment without tabs</li> <li>- "General" tab on tabs</li> </ul>	<ul style="list-style-type: none"> <li>- Sequence only</li> </ul>

Interface Design – Overview

Customer enhancement CEI00000	List	Yes	<ul style="list-style-type: none"> <li>- Move characteristics and values</li> <li>- Define border</li> <li>- Screen size of list</li> </ul>	<b>Function description:</b> The interface design lets you assign the attribute "pushbutton" to a characteristics group. If the name of the characteristics group starts with CUSTOMER_PUSHBUTTON, a CUSTOMER FUNCTION (user-defined function) is processed when you choose the pushbutton.
----------------------------------	------	-----	---	--

**For materials only:** for the **sequence**, you can define that the characteristics group is only displayed in sales or in engineering (one sequence per application). You can select an indicator for sales (sales order and configuration simulation) or engineering (order BOM and configuration simulation). If you do not select an indicator, the interface design is valid for both sales and engineering.

## Result

If the characteristics group is summarized, the characteristic no longer appears on the general tab.

You can now display the characteristics group in the selected interface design.

**For more information on grouping characteristics, creating interface designs, and defining sequences for characteristics, see:**

[Defining an Interface Design \[Page 248\]](#)

[Maintaining an Interface Design \[Page 250\]](#)

[Defining the Sequence of Characteristics \[Page 252\]](#)

## Defining an Interface Design

# Defining an Interface Design

## Prerequisites

- Your user master record contains authorization object C\_LOVC\_DSG with the activity that you want to perform.
- The configuration profile contains a name for an interface design under *Config initial screen* → *UserInterf*. This activates the function on the value assignment screen.

## Procedure

### Creating an Interface Design

1. In the configuration editor (value assignment screen) choose *Value assignment* → *Interface design* → *Characteristic grouping*.

You see a dialog box in which you must enter a name for the group. The characteristics are grouped under this name.

- You can define whether the groups of characteristics are displayed as pushbuttons or tabs on the value assignment screen. You can define up to 15 pushbuttons and up to 10 tabs.
- If you do not want to display the characteristics individually on the value assignment screen, select *Summarize*.

Tabs are summarized automatically.



The *Summarize* function does not apply to the *Sequence* indicator.

Confirm.

2. You see the processing screen, where you can define the user interface for the grouped characteristics.
3. Choose *Characteristics*. Select the characteristics that you want to group together.
4. Save your interface design.

### Changing an Interface Design

1. In the configuration editor (value assignment screen), choose *Value assignment* → *Interface design* → *Overview*, and call the characteristics group you want to change.
2. You see the processing screen for interface design, where you can make your changes.
3. Save your interface design.

### Deleting an Interface Design

1. Call the characteristics group that you want to delete and go to the processing screen for interface designs.
2. Choose *Goto* → *Chars group* on the processing screen for interface design.
3. You can now delete the characteristics group and save.





## Maintaining an Interface Design

### Prerequisites

Your user master record contains authorization object C\_LOVC\_DSG with the activity that you want to perform.

The configuration profile contains a name for an interface design under *Config initial screen* → *UserInterf*. This activates the function on the value assignment screen.

### Procedure

Go to the characteristic value assignment screen by choosing *Values* → *Interface design* → *Overview*, then select a characteristics group.

### Settings for Characteristics/Characteristic Values

Choose *Settings* to define whether characteristics and value are processed separately or together (when you move them, for example).



If you use tabs or sequences, you cannot separate characteristics from their values.

### Move

You can move characteristics and values:

1. Select the characteristic or value you want to move. Select one line only.
2. Position the cursor where you want to move the characteristic or value to.
3. Choose *Move*. The characteristic or value moves to the new position.

### Defining a Border (Only Applies to Characteristics Groups and Pushbuttons)

You can define a border for characteristics or values. To do this:

1. Place the cursor on the top left-hand corner of the border.
2. Choose *Border*.
3. Enter the border name. You can also enter language-dependent descriptions for the border by choosing *Descriptions*.
4. Confirm your entries.

You see a message telling you to select the lower edge of the border. Confirm this message.

5. Place the cursor on the bottom right-hand corner of the border.
6. Choose *Select*.

The characteristics have a border with the description you entered.

To delete a border, select the border with the cursor and choose *Delete*.

### Delete Line/Insert Line

To delete a line, choose *Edit* → *Delete*. To insert a line, choose *Edit* → *Insert*. Inserted lines always appear above the line you select.

### Add/Delete Characteristics

Choose *Characteristics* and select the characteristics you want to add or delete.

## Defining the Sequence of Characteristics

# Defining the Sequence of Characteristics

## Use

If you have to assign values to a large number of characteristics, it is useful to define a sequence for the configuration editor. This lets you prioritize the characteristics.

You change the sequence of characteristics by choosing *Value assignment* → *Interface design*, not on the value assignment screen itself.



This function only applies to the *General* tab and the standard settings.

## Prerequisites

Your user master record contains authorization object C\_LOVC\_DSG with the activity that you want to perform.

The configuration profile contains a name for an interface design under *Confign initial screen* → *UserInterf*. This activates the function on the value assignment screen.

## Procedure

1. Choose *Value assignment* → *Interface design* → *Characteristic grouping* on the value assignment screen.

You see a dialog box in which you enter a name for the characteristics group.

- Set the *Sequence* indicator.



You cannot set the *Sequence* indicator if the *Tabs* indicator or the *Pushbutton* indicator is set.

Confirm.

2. You see the processing screen for interface design.

3. Choose *Characteristics*.

Select the characteristics in the sequence that you want on the value assignment screen.

- a. Select the characteristic that you want to display at the top, and confirm.
- b. You see the characteristic on the processing screen.
- c. Call the dialog box again and select the characteristic that you want to display next.
- d. Confirm.

The characteristic is displayed below the first characteristic you selected.

- e. Continue in the same way until you have all characteristics in the sequence you require.

4. Save your settings.

---

Defining the Sequence of Characteristics

## Result

The sequence is copied to the value assignment screen.

To change the sequence, choose *Value assignment* → *Interface design* → *Overview* and call the characteristics. You can move or delete characteristics. To delete the sequence, choose *Goto* → *Chars group* on the *Overview* screen. You see a dialog box where you delete the sequence.

## Relevance to Printing of Characteristics

### Use

You can use the interface design to define characteristics as relevant to printing for sales or purchasing. In the print functions in these applications, only the characteristics selected in the characteristics group are printed.

### Integration

This function is integrated in the print functions for purchasing and sales.

When you print documents in sales (order confirmations, deliveries, and billing documents) or in purchasing (purchase orders), configurable items and material variants are printed with their characteristics and the values assigned to the item.

### Prerequisites

Your user master record contains authorization object C\_LOVC\_DSG with the activity that you want to perform.

The configuration profile contains a name for an interface design under *Confgn initial screen* → *UserInterf*. This activates the function on the value assignment screen.

### Activities

1. Select the characteristics that are relevant to printouts.
2. Choose *Value assignment* → *Interface design* → *Characteristic grouping*.
3. Enter a name for your characteristics group.  
Enter the application where the characteristics are relevant to printing.
4. In the [interface design \[Page 250\]](#) for characteristic value assignment, you can define how the characteristics are displayed.
4. Save your characteristics group.



If no characteristics group is maintained for a print-relevant item, all characteristics are printed with their assigned values. However, if a characteristics group for printouts exists, only the characteristics assigned to the group are printed in the sequence you define. In both cases, only characteristics with assigned values are printed. Characteristics that are hidden by object dependencies (such as structure SCREEN\_DEP) are also printed, if they have assigned values.

## Specifying Enhancements in the Configuration Editor

### Use

You can use enhancements in the configuration editor to call your own interfaces and functions to complete the configuration process.

In enhancement CEI0000, there are 10 function exits, which you can fill out with your own functionality. To call function exits, you choose pushbuttons that you define in the interface design function.

### Procedure

1. Choose *Value assignment* → *Interface design* → *Characteristic grouping*.
2. Enter a *Characteristics group* name that starts with:

CUSTOMER\_PUSHBUTTON\_<xxx>

- Select *Pushbutton* as the display option.
- Enter a language-dependent description and a name for the pushbutton.

#### Example:

Language	Description	Pushbutton
EN	CUSTOMER_PUSHBUTTON_XYZ	Function1

3. On the interface design processing screen, choose *Save* without making any other entries.

You see the pushbutton on the value assignment screen with its name (for example, Function1).

4. To execute the function exit, you choose the pushbutton.

In the standard system, you can enter up to 10 pushbuttons on the value assignment screen, so you can execute any one of 10 function exits.

The pushbuttons and function exits are assigned to each other as follows:

Pushbutton 01	EXIT_SAPLCEI0_010
Pushbutton 02	EXIT_SAPLCEI0_011
Pushbutton 03	EXIT_SAPLCEI0_012
Pushbutton 04	EXIT_SAPLCEI0_013
Pushbutton 05	EXIT_SAPLCEI0_014
Pushbutton 06	EXIT_SAPLCEI0_015
Pushbutton 07	EXIT_SAPLCEI0_016
Pushbutton 08	EXIT_SAPLCEI0_017
Pushbutton 09	EXIT_SAPLCEI0_018

---

**Specifying Enhancements in the Configuration Editor**

Pushbutton 10	EXIT_SAPLCEI0_019
---------------	-------------------

The first pushbutton you create is automatically assigned internally to the first function exit, the second pushbutton is assigned to the second function exit, and so on.

## Integration

The function exits are called in program LCEI0F01 in FORM routine EXECUTE\_PUSHBUTTON\_GROUP.

## Prerequisites

The configuration profile contains a name for an interface design. This activates the function on the value assignment screen.

You have included the function exits predefined by SAP in your program code as include programs. Use transaction CMOD to do this. For more information, see the application help on transaction CMOD.



## The Configuration Result

### Definition

The configuration result shows the BOM of the configurable material according to the settings in the configuration profile.

### Use

You can display the configuration result if *Result* is selected as an allowed screen in the configuration screen.

You can display the configuration result in the configuration simulation, in sales documents, and when processing the order BOM. The BOM explosion in the sales order depends on the settings in the configuration profile:

Configuration Profile	Sales Orders	Simulation	Order BOM Processing
No BOM explosion	No	No	-
Single-level BOM explosion	Yes	Yes	Yes
Multi-level BOM explosion	Yes	Yes	Yes
Sales order	Yes	<i>Sales</i> setting	-
Order BOM	No	<i>Engineering</i> setting	Yes

- With the setting *BOM explosion: None*: means that components of the BOM are not determined until the material requirements planning (MRP) stage. Do not use this setting with *Process order BOM*.
- You cannot select processes *Sales order* and *Order BOM* at the same time. This means that you cannot process an order BOM if you select *Sales order*.

### Non-Variable Parts and Variant Parts

In the configuration result of a BOM, you see the components that were selected according to the characteristic values assigned. These include:

- Non-variable parts
 

These components are required in all variants of the product. They are not dependent on specific characteristic values.
- Variant parts
 

These components are not always required in configuration. They are only required for specific variants of the product. Selection conditions define when a component is included in the configuration.

### Changes to BOM Items

BOM items can be changed either manually or by the system:

### The Configuration Result

- You can make changes manually if you maintained the appropriate setting in the configuration profile for the configurable material (*Order BOM* or manual changes allowed plus *Process: Sales order*).
- The system changes BOM items if the entry in an item field is changed by dependencies (see [Using Fields with a Reference to Master Data in Configuration \[Page 96\]](#)).

### Configuration is Active on Request

You can configure on request on the result screen. choose *Configuration* → *Configurator* and select *Active on request*. If the configuration is not active, dependencies are only processed when you choose *Configure*, and system performance improves.

### Integration

You can display the configuration result in a simulation, when you configure a product in a sales order, and when you process an order BOM.

**See also:**

[Controlling the BOM Explosion \[Page 44\]](#)

## Explanation Functions: Result Screen

You can display the following BOM item information on the result screen:

### Non-Variable Parts and Variant Parts

You can whether the item is a non-variable part or a variable part by choosing *Information* and *Explanation*.

You can display a dependency for a variant part by choosing the *Explanations*.

### Changed BOM Items

BOM items that have been changed by dependencies have an information icon next to them.

To see which changes were made to the item and display the dependencies, choose *Information*.

### Replacing a Class Item

BOM items that have replaced class items have an information icon next to them.

To see which class was replaced, choose *Information*.

### Classification Data

To display the classification data of a material in the configuration, choose *Classification*. In BOM maintenance, you must enter the class in which the material is classified in the item, by choosing *Goto* → *Item* → *General data* (see [Classification as a Selection Condition \[Page 15\]](#)).

### Analysis Tool

To display the data of the configuration model, choose *Extras* → *Analysis*. You see the following information:

- Number of configurable items and class nodes
- Total number of characteristics, number of required characteristics, number of characteristics with assigned values, and number of restrictable characteristics
- Total number of values, number of valid values, number of default values, and number of assigned values
- Overview of dependencies processed
- Exploded BOM with dependencies for BOM items
- Class nodes
- List of characteristics and values of configurable items

## Defining Scope and Display Options for the Result

## Defining Scope and Display Options for the Result

## Use

## Display Options for BOM

You can define how a multi-level BOM is displayed in the configuration result when you explode the BOM:

- The individual levels are exploded after each other:
  - Comp-1
  - Comp-2
  - Comp-3
  - 
  - Comp\_2-1
  - Comp\_2-2
  - Comp\_2-3
  - 
  - Comp\_2-2-1
  - Comp\_2-2-2
- The assemblies are exploded as and when they occur:
  - Comp-1
  - Comp-2
    - Comp\_2-1
    - Comp\_2-2
      - Comp\_2-2-1
      - Comp\_2-2-2
    - Comp\_2-3
  - Comp-3

## Special Procurement Key for Phantom Assemblies

You can define whether special procurement key 50 (phantom assembly) is read.

Special procurement key 50 is assigned in the MRP data of a material master record. This key means that components of these assemblies are grouped together for logical or logistical reasons, but the assembly is not actually assembled.

- If the special procurement key is read, the phantom assembly is not displayed. The components of the phantom assembly are exploded below the next assembly up.
- If the special procurement key is ignored, the phantom assembly is displayed.

(See the R/3 Library *PP Material requirements planning (MRP)* [Phantom Assemblies \[Ext.\]](#))

---

Defining Scope and Display Options for the Result

## Scope of the BOM Explosion

You can define the following settings for the BOM explosion:

- The characteristics of configurable assemblies are shown.
- The dependencies for variant parts are shown.
- Only assemblies and class items are shown.
- Only configurable assemblies are shown.
- Components are displayed whether they have been selected or not.

Components that have not been selected are checked with an x.

## Procedure

1. On the result screen, choose *View* → *Display options*.

You can define display options and settings for the special procurement key.

2. On the result screen, choose *View* → *Scope*, to define the scope of the BOM explosion.



The settings defined in the configuration profile or in the *Settings* on the value assignment screen do not influence the result screen.

## Configurable Materials in Sales Documents

### Use

Configurable materials are complex products that can be produced in different variants. For example, cars have different paintwork and different-sized engines. You use characteristics to describe configurable materials.

There are different ways of processing configurable materials in sales documents. These processing options are described in the following scenarios:

- [No BOM Explosion \[Page 46\]](#)
- [Single-Level BOM Explosion \[Page 49\]](#)
- [Multi-Level BOM Explosion \[Page 51\]](#)
- [Process: Sales Order \[Page 37\]](#)
- [Process: Order BOM \[Page 42\]](#)

You can configure configurable materials in quotations and sales orders. You can copy the configuration from a quotation to a sales order.

### Prerequisites

Before you can configure a material in a sales document, the following prerequisites must be fulfilled:

#### Settings for Variant Configuration

- You have created the material with a material type that supports variant configuration or defined the material as configurable in its material master record.  
  
You have created sales data in the material master of the material (see [Material Master Data for Configurable Materials \[Page 12\]](#)).
- You have assigned characteristics to the material by allocating the material to a variant class (see [Classification \[Page 32\]](#)).
- You have created a configuration profile for the material, containing at least the configuration parameters that are essential for controlling processing in sales documents (see [The Configuration Profile \[Page 27\]](#)).
- You have created dependencies and allocated these to characteristics and characteristic values, to define the dependencies between characteristics and values.  
When you configure a material, some characteristic values tend to be mutually exclusive. This may be for technical reasons (for example, certain engines can only use a certain type of transmission) or for marketing reasons (for example, leather covers are only available with more expensive models). See [Object Dependencies \[Page 65\]](#).
- You have created selection conditions and allocated these to items in BOMs and operations in routings. Selection conditions ensure that the system selects the items and operations you need to produce your variant (see [Selection Conditions \[Page 77\]](#)).

## Configurable Materials in Sales Documents

### Settings in the Sales

- The item category of the order item must allow configuration. In Customizing for *Sales and Distribution*, the correct indicator must be set for the item categories in the *Structure scope* field.  

The item category group entered in the material master controls which item categories can be assigned.
- The *Configuration allowed or required* indicator must be set for the requirements class in Customizing for *Sales and Distribution*.  

The requirements class is determined using the strategy group and requirements type defined in the material master.

### Process Flow

1. If you enter a configurable material on the item entry screen, then confirm, you see the configuration editor with the characteristics of the configurable material. You configure the material by assigning values to the characteristics (refer to [Configuring Objects \[Page 240\]](#)).
2. Once you have finished configuring the configuration structure, go back to the sales document.
3. You see the result of the availability check. This tells you whether the requested delivery date is feasible. If possible, the system offers a delivery proposal (see the R/3 Library *Sales → Availability Check and Requirements in Sales and Distribution Processing → Availability Check in Sales and Distribution Processing → Working with the Availability Check in Sales and Distribution Processing ### Reactions to the Availability Check in Sales Documents*).  

Confirm a delivery date or select *Continue*.
4. You see the order item overview screen again.
5. You can continue processing the sales order.
6. Once you have finished processing, save your sales order.

### Result

The configuration of the material is saved with the sales order. Customer requirements are transferred to MRP.

## Item Categories for Configurable Materials

## Item Categories for Configurable Materials

In the standard R/3 System, the following item category groups determine which item categories are assigned to configurable materials in the sales order:

- Item category group 0002  
Pricing and transfer of requirements for header material (refer to [Graphic 1 \[Page 266\]](#))
- Item category group 0004  
Pricing and transfer of requirements for components (refer to [Graphic 2 \[Page 267\]](#))

The BOM of the configurable material may contain configurable assemblies, which are copied to the sales order as subordinate items. The item category group of the materials and the item category of the superior material determine the item category of these subordinate items.

### Determining Item Categories

	Superior configurable material	Non-configurable assembly	Configurable assembly
Item category group	0002	NORM	0002
Item category	TAC	TAE	TAE

Refer to: [Graphic 3 \[Page 268\]](#)

	Superior configurable material	Non-configurable assembly	Configurable assembly
Item category group	0002	NORM	0002
Item category	TAE	TAE	TAE

Refer to: [Graphic 4 \[Page 269\]](#)

	Superior configurable material	Non-configurable assembly	Configurable assembly
Item category group	0002	NORM	0004
Item category	TAC	TAE	TAE

Refer to: [Graphic 5 \[Page 271\]](#)

	Superior configurable material	Non-configurable assembly	Configurable assembly
Item category group	0004	NORM	0002
Item category	TAM	TAN	TAC



Item Categories for Configurable Materials

Refer to [Graphic 6 \[Page 270\]](#)

	Superior configurable material	Non-configurable assembly	Configurable assembly
Item category group	0004	NORM	0004
Item category	TAM	TAN	TAM

Refer to [Graphic 7 \[Page 272\]](#)

**See also:**

R/3 Library *SD Sales* [Item Categories \[Ext.\]](#)

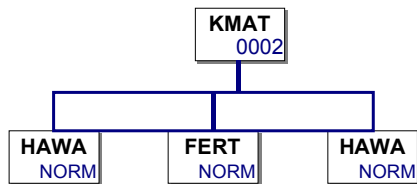
## Graphic 1

## Graphic 1

## Requirements Transfer and Pricing at Header Level

The header material has item category group **0002**. The BOM for the material is made up of non-configurable finished products and trading goods. These materials have item category group **NORM**:

- In the sales order, the superior material has item category **TAC**. Items of this category transfer requirements and have an individual price.
- The subordinate order items have item category **TAE**. These items are not relevant to transfer of requirements or pricing, and appear in the sales order for information purposes only.



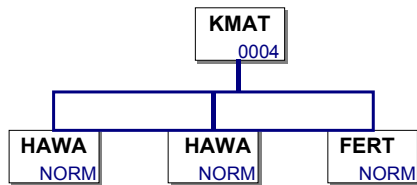
Sold-to party		33		
Requested delivery date		07/01/99		
10		KMAT	3 PC	TAC
20	(10)	HAWA	3 PC	TAE
30	(10)	FERT	6 PC	TAE
40	(10)	HAWA	6 PC	TAE

## Graphic 2

### Requirements Transfer and Pricing at Assembly Level

The header material has item category group **0004**. The BOM for the material is made up of non-configurable finished products and trading goods. These materials have item category group **NORM**.

- In the sales order, the configurable material has item category **TAM** and is not relevant to pricing or transfer of requirements.
- The subordinate order items have item category **TAN**. Order items with this item category transfer requirements and are relevant to pricing.

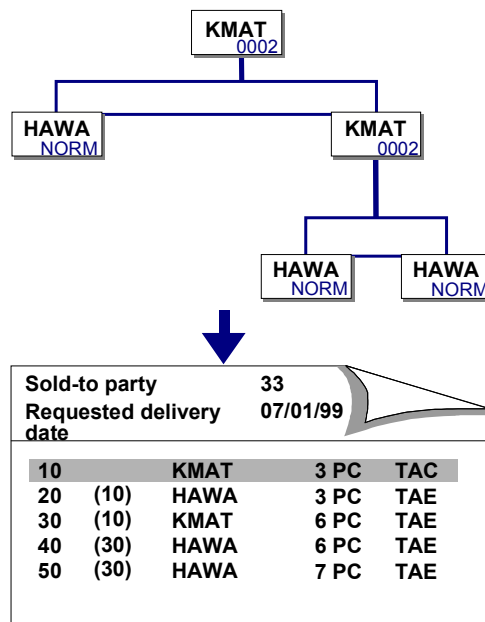


Sold-to party		33		
Requested delivery date		07/01/99		
10		KMAT	3 PC	TAM
20	(10)	HAWA	3 PC	TAN
30	(10)	FERT	6 PC	TAN
40	(10)	HAWA	6 PC	TAN

Graphic 3

## Graphic 3

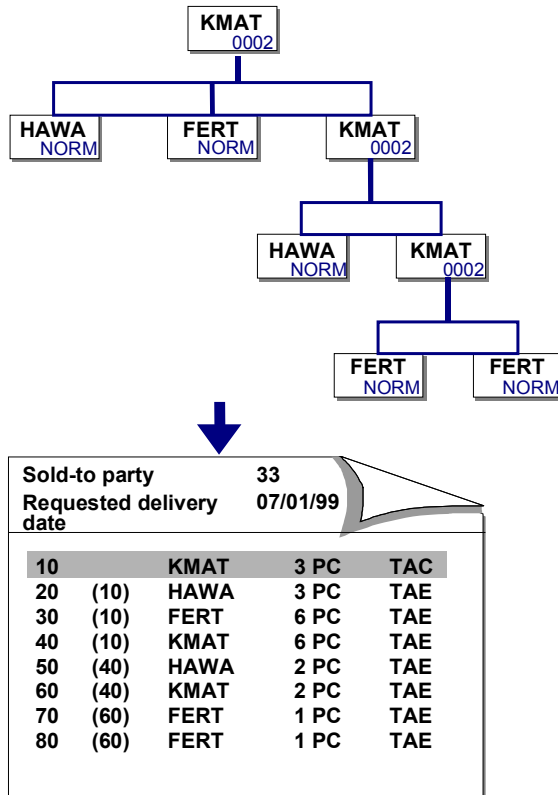
If the superior configurable material has item category **TAC**, a subordinate configurable material with item category group **0002** has item category **TAE**. This means that the material is not relevant to pricing and does not transfer requirements. Components with item category group **NORM** also have item category **TAE**.



## Graphic 4

If the immediately superior configurable material has item category **TAE**, a subordinate configurable material with item category group **0002** has item category **TAE**. This means that the material is not relevant to pricing and does not transfer requirements.

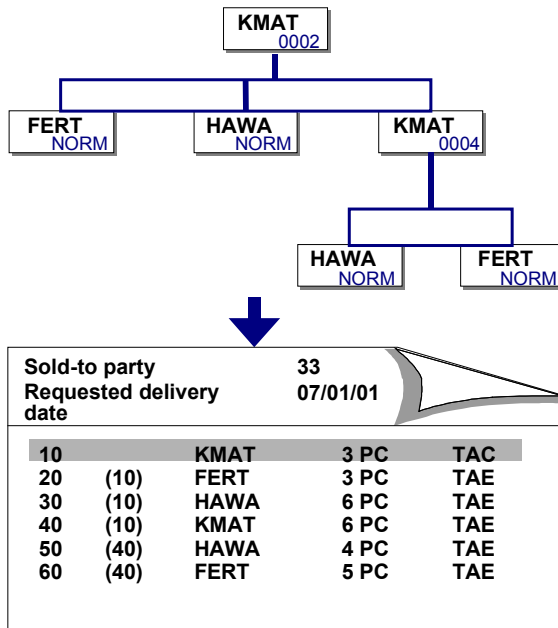
Components with item category group **NORM** also have item category **TAE**.



Graphic 5

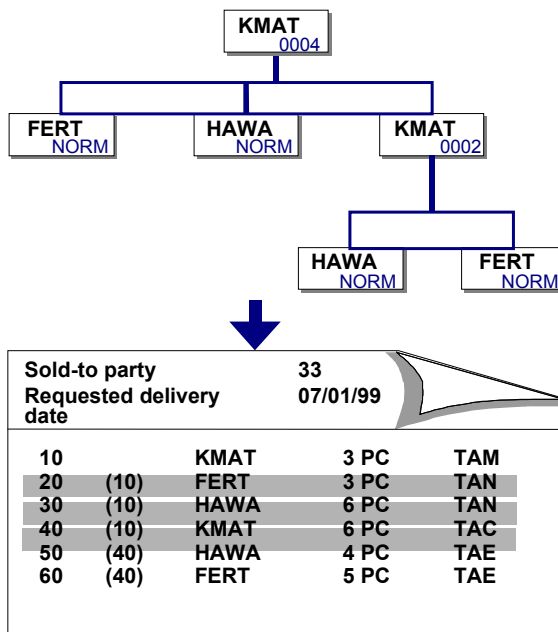
## Graphic 5

If the superior configurable material has item category TAC, a subordinate configurable material with item category group 0004 has item category TAE. Components with item category group NORM also have item category TAE. Only the header material is relevant to pricing and transfers requirements.



## Graphic 6

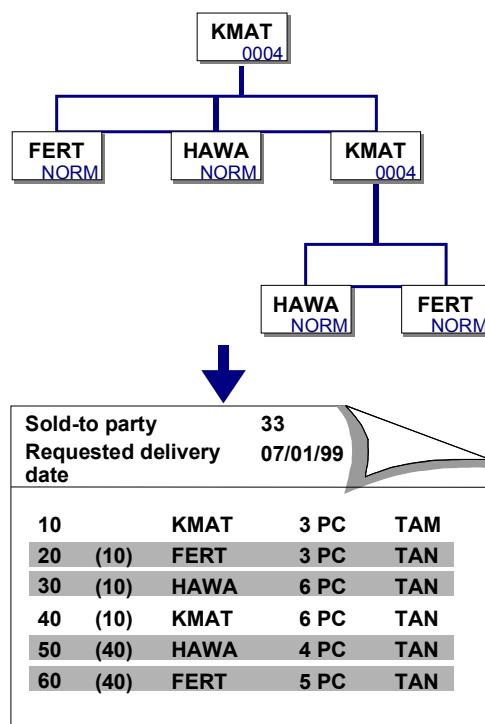
If the superior configurable material has item category **TAM**, a subordinate configurable material with item category group **0002** has item category **TAC**. This means that the subordinate material transfers requirements. Components of the header material have item category group **NORM** and item category **TAN**, and do not transfer requirements. Components of subordinate configurable materials with item category group **NORM** have item category **TAE** and do not transfer requirements.



Graphic 7

## Graphic 7

If the superior configurable material has item category **TAM**, a subordinate configurable material with item category group **0004** also has item category **TAM**. Components with item category group **NORM** have item category **TAN** and are relevant to transfer of requirements and pricing.





## Variant Matching in the Sales Order

### Use

If material variants already exist for a configurable material, the configurable material can be replaced in the sales order by a material variant. For this to happen, the configuration of the material must match the material variant exactly. Material variants that only partly match the configuration of the material in the sales order are ignored.

### Prerequisites

You define the settings for variant matching in Customizing for Sales and Distribution, where you maintain item categories.

- You activate variant matching for an item category
- You define how the system reacts if it finds a material variant:
  - The system automatically replaces the configurable material with the material variant.
  - You see a message telling you that a material variant with the same configuration exists.
- You determine whether you want an availability check to be executed.

If the material variant is not available, you can:

- Receive an automatic message with a list of the material variant stock situation. You can then decide whether to replace the material variant.
- Decide that you do not want to replace the variant.

### Restrictions

You can only replace materials that have the configuration parameter *BOM explosion: None* with material variants.

### Activities

1. Configure the configurable material in the sales order.
2. On the value assignment screen, you can check whether material variants match the configuration of the material (refer to [Variant Matching on the Value Assignment Screen \[Page 60\]](#)).  
Before the material can be replaced in the sales order by the material variant, the configurations of the materials must match completely.
3. Once you have configured the material, leave the configuration editor.
4. Depending on the settings in Customizing, either the material is replaced by the material variant immediately, or you see a message telling you that a suitable material variant has been found.
5. If the material was replaced, the order item contains the material variant.

---

**Variant Matching in the Sales Order**

Individual or collective requirements can be generated for the material variant. Pricing also applies to the material variant, unless you define the configurable material as the pricing material for the material variant.

**See also:**

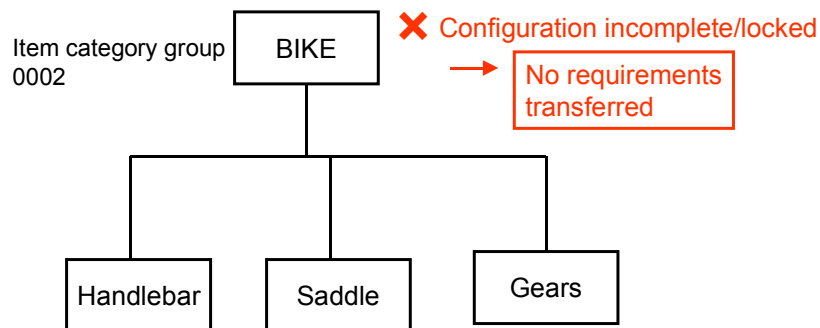
[Material Variants \[Page 224\]](#)

## Transfer of Requirements for Locked Configurations

### Use

In the standard system, no requirements are transferred for sales order items with a locked or incomplete configuration status.

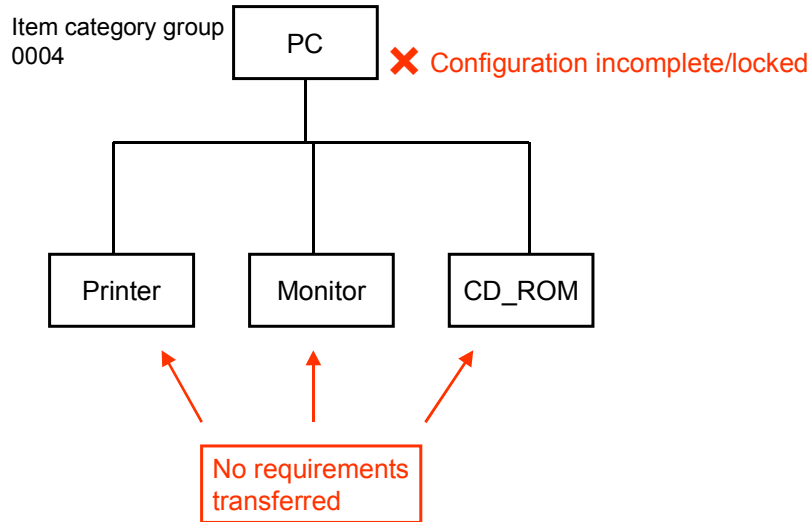
#### Requirements Transfer for Main Item



If the header material has a locked or incomplete configuration status, components that would normally transfer requirements lose their relevance to requirements transfer.

## Transfer of Requirements for Locked Configurations

## Requirements Transfer for Components



No planned order is generated in material requirements planning (MRP) for the order item.

## Features

**System Status *Configuration Incomplete***

If the configuration of a sales document item is incomplete, the item has system status KONU (configuration incomplete).

**Business Transaction Requirements *Transfer with Incomplete Configuration***

Business transaction *Incomplete req. configuration* controls transfer of requirements. In the standard system, no system status allows this business transaction. This means that requirements for the main order item and its components lose their relevance to MRP.

**User Status**

In Customizing for *Sales and Distribution*, you can define a user status that allows this business transaction. This status can be set manually or by setting it as the initial status in the status profile. You can use it to ensure that requirements are transferred for order items, even if they have locked or incomplete configuration status.

In Customizing for *Sales and Distribution*, you can define a user status and assign it to a status profile, by choosing *Sales* → *Sales Documents* → *Define and assign status profile*. You can define a user status so that it allows business transaction *Incomplete req. configuration*. You can define this as the initial status, so that it is always set for the status profile.

You assign the status profile to an item category.



## Transfer of Requirements for Locked Configurations

You can use a user status to prevent further changes to the configuration data of an order item. Business transaction *Change configuration data* is defined for this purpose. You can use a user status to allow or disallow this transaction.

### Activities

The configuration status of a configurable material is defined in the configuration editor by choosing *Extras* → *Change status*. Status *Incomplete* is set automatically if any required characteristics do not have assigned values.

In the sales document, the system sets system status KONU for the order item that has an incomplete or locked configuration.

1. To display the status, select the item and choose *Goto* → *Item* → *Status*.

If a status profile is defined with an initial status, you see the user status.

2. For more detailed information, choose *Object status*.

You see the active system and the user statuses in the status profile. You can set a user status manually.

You can display the allowed and disallowed business transactions:

- With a user status, transaction *Incomplete req. configuration* is not allowed.
- If the status profile has a user profile that allows the transaction, and this status is set, the transaction is allowed.

## Changing Fields in a Sales Order

## Changing Fields in a Sales Order

### Use

You can use reference characteristics to access the following fields in a sales order:

- VBAK Header data
- VBAP Item data
- MAEPV Material master fields
- MAAPV Material master fields
- VBKD Sales document: commercial data
- VBPA\_AG Partner: sold-to party
- VBPA\_WE Partner: ship-to party
- VBPA\_RE Partner: bill-to party
- VBPA\_RG Partner: payer



However, you can only change the item data. Access to other fields is read only. This data is required as information for dependencies, for example.

The fields that you can change are in structure VCSD\_UPDATE:

VCSD_UPDATE	Gross weight of item (BRGEW)
	Net weight of item (NTGEW)
	Unit of weight (GEWEI)
	Volume (VOLUM)
	Volume unit (VOLEH)
	Item quantity (KWMENG)
	Sales unit of measure (VRKME)
	Conversion factors
	Target quantity (ZMENG)
	Target quantity unit (ZIEME)
	Article description (ARKTX)



Only *configurable* materials and components are supported.

## Changing Fields in a Sales Order

You can use the following customer functions to add further fields from table VBAP to structure VCSD\_UPDATE and transfer the changed values from the configuration to the sales order:

- EXIT\_SAPLCEI0\_001 (include fields in structure VCSD\_UPDATE)
- EXIT\_SAPFV45S\_002 (process changed field values)

## Process Flow

1. In characteristics maintenance, create a reference characteristic that refers to a field in structure VCSD\_UPDATE.
2. Assign the reference characteristic to the variant class.
3. Create a procedure that uses the reference characteristic to refer to the table.
4. Assign this dependency to the configuration profile of the configurable material.
5. When you configure the item, the changed values (for example, the weight) are automatically transferred to the sales document and set.

If a value set by the configuration process cannot be transferred to the document, you see a system message.



The values set during the configuration process can only be overwritten in the sales order if the following requirements are fulfilled:

- The value of the reference characteristic was set by a default value or a user entry.
- The configuration is changeable in the sales document.

### See also:

[Reference Characteristics in Dependencies \[Page 98\]](#)

## Low-Level Configuration

# Low-Level Configuration

R/3 Variant Configuration distinguishes the following types of configuration:

- High-level configuration
- Low-level configuration

## High-Level Configuration

High-level configuration is for an interactive configuration task, in the sales order, for example. During configuration, the user assigns values to characteristics. Dependencies between configurable materials in the configuration structure can be mapped using constraints.

## Low-Level Configuration

Low-level configuration is for non-interactive configuration. Low-level configuration refers to “background” explosions of bills of material (BOMs), routings, maintenance orders, model service specifications, and standard networks, for example, in material requirements planning (MRP), or when creating production orders or networks. Characteristic values from the sales order, for object \$ROOT or \$PARENT, are automatically used to determine the BOM components and operations or activities.

The following dependencies are read:

- Selection conditions for components and operations
- Procedures and actions for changing field values in master data fields



We recommend that you **always** use **procedures** to change master data in BOM items, not actions.

Classification data can also be used.

- Class items can be replaced by one component (not several).
- The classification of a component, operation, or activity can be used as a selection condition.

The following restrictions apply to low-level configuration:

- Dependencies for characteristics, characteristic values, and configuration profiles are ignored.
- Constraints are not processed in low-level configuration.
- The expressions \$SET\_PRICING\_FACTOR, \$SET\_DEFAULT, \$DEL\_DEFAULT, \$SUM\_PARTS, \$COUNT\_PARTS, and TYPE\_OF are not supported.



## Configurable Materials in Purchasing

### Purpose

You can use variant configuration for purchasing materials. This means that the material is procured externally, rather than produced in-house.

You can create and change the configuration of configurable materials in the following purchasing documents: purchase requisitions, requests for quotation, purchase orders, and outline agreements.



Subcontracting items and archiving of characteristic values are not supported.

If no configuration exists of a configurable material, you can *create* the configuration in the purchasing document. This may be necessary if, for example, you want to order a configurable material without a reference to a sales order.

If the configuration of the configurable material has been copied from a sales order or the material master, you can *change* it in the purchasing document. This may be necessary if the material cannot be supplied in the selected configuration, and has to be adjusted to match the quotation. In this case, the price is recalculated.



If you change the configuration of a material in the purchasing document, this does **not** change the original configuration in the sales order or material master.

If you make subsequent changes in the sales order or material master, these do **not** affect the configuration in the purchasing document.

### Procedure (Example for a Purchase Requisition)

1. Create a sales order containing a configurable material that is procured externally. The material need not have a BOM. However, the material must have a configuration profile with an assignment to a class. The characteristics of the class are used to describe the material.

Your externally procured material may be part of a BOM. Depending on the BOM explosion settings in the configuration profile, the material may be configured either independently or according to the characteristic values assigned to the BOM header material.

2. A purchase requisition can be generated either in the sales order itself or as a result of a material requirements planning (MRP) run. The purchase requisition contains the characteristic values that were assigned to the material in the sales order.
3. The purchase requisition is converted to a purchase order. The purchase order contains the characteristic values that were assigned to the material in the sales order. When you print the purchase order, these characteristic values are printed with it.

**See also:**

[Example: Configurable Materials in Purchasing \[Ext.\]](#)

[Variant Conditions in Purchasing \[Page 212\]](#)



## Displaying a Configuration Overview

You can display the variants of a configurable object. For example, a material could already be configured as a sales order, a production order, or a material variant.

1. From the variant configuration menu, choose Environment → Configured objects, and select the object type for the configuration overview.
2. Enter the object key and choose *Configured objects*.
3. If different types of variant exist, select the type of variant you want to see.  
If only one type of variant exists, such as configured sales orders, you see a list of objects immediately.
4. You can select a variant from the list and display its characteristic values.  
You can also print the list and find a specific variant in the list.

## Enhancements in Variant Configuration

## Enhancements in Variant Configuration

### Use

In order to optimize the business processes in variant configuration, you can change some variant configuration functions of the SAP System by using customer exits.



If a user works with very complex, multilevel configurations, you can control the level of detail shown for the configuration. You can use enhancement CCUX0800 to determine whether all assemblies are exploded or only the configurable assemblies. The enhancement contains function module EXIT\_SAPLCUKO\_002. This links to include program ZXCUCU05. Enter your program code in this include.

#### See also:

[Enhancements to the SAP System in the Area of PDM \[Ext.\]](#)

[Customer Exits \[Ext.\]](#)

### Prerequisites

When you use a function exit, you must create a company-specific include program that matches the programming logic of the function module.

### Features

The following is an overview of the enhancements supported in variant configuration (development class CU).



Constant additions are being made to customer enhancements for variant configuration and this documentation may not include all the possible enhancements. To see the most up-to-date list of enhancements call the F4 help for *Enhancements* and the *Development class CU\** for *Infosystem*.

Description	Enhancement	Function Modules	Include
Variant configuration: external APIs	CAVC0000	EXIT_SAPLCAVC_CFG_001 EXIT_SAPLCAVC_INST_001 EXIT_SAPLCAVC_INST_002	ZXCAVCU01 ZXCAVCU02 ZXCAVCU03
Customer-specific batch-input processing	CCUCEI0B	EXIT_SAPLCEI0_020	ZXCEI0U12
Processing of planning tables	CCUP0001	EXIT_SAPLCUD2_800 EXIT_SAPLCUTS_800	ZXCUPU02 ZXCUPU01
Additional checks on configurations	CCUX0000	EXIT_SAPLCUKO_001	ZXCUCU02
Functions for loading configurations	CCUX0001	EXIT_SAPLCUD0_001 EXIT_SAPLCUXC_001	ZXCUCU01 ZXCUCU03


Enhancements in Variant Configuration

Reaction to conflict when finding an object for a class node	CCUX0002	EXIT_SAPLCUD0_002	ZXCUCU04
Parameters for finding an object for a class node	CCUX0003	EXIT_SAPLCEIS_001	ZXCUCU17
Postprocessing of configuration with object dependencies	CCUX0004	EXIT_SAPLCUKO_003	ZXCUCU07
Transfer of item category after material variant matching	CCUX0005	EXIT_SAPLCEB1_001	ZXCUCU10
Fixing an order BOM	CCUX0006	EXIT_SAPLCUKO_007	ZXCUCU13
Definition of the BOM category for instantiation	CCUX0007	EXIT_SAPLCEB1_002	ZXCUCU14
No BOM explosion for externally procured components	CCUX0008	EXIT_SAPLCUKO_008	ZXCUCU15
Synchronization of initialization of variant configuration	CCUX0100	EXIT_SAPLCUD0_003	ZXCUIU01
Configuration: Additional processing for changing variant table contents	CCUX0510	EXIT_SAPLCUD3_001	ZXCUTU02
Effectivity date for order BOM	CCUXDATE	EXIT_SAPLCASL_002	ZXCUC1U03
Control of the level of detail in multilevel configurations	CCUX0800	EXIT_SAPLCUKO_002	ZXCUCU05
Explosion date for result-oriented order BOMs	CCUXDATU	EXIT_SAPLCASL_001	ZXCUC1U01
Maintenance of additional data for instantiation	CCUXIACD	EXIT_SAPLCEB1_100	ZXCUC1U02
Effectivity date for order BOM	CCUXDATE	EXIT_SAPLCASL_002	ZXCUC1U03
Modification for external number assignment for instantiation	CCUXINST	EXIT_SAPLCUKO_004	ZXCUCU09
Find material variants with the same value assignment	CCUCEI0V	EXIT_SAPLCEI0_023	ZXCI0U15
Assigned values file and object characteristics	CCUCEI0A	EXIT_SAPLCEI0_021	ZXCEI0U13
Change F4 help for characteristics in configuration	CCUCEI0H	EXIT_SAPLCEI0_022	ZXCEI0U14
Object types for finding objects for class nodes	CCUXOBTY	EXIT_SAPLCEIS_002	ZXCUCU19

**Enhancements in Variant Configuration**

Definition of the BOM status for instantiated materials	CCUXSTAT	EXIT_SAPLCEB1_003	ZXCUCU16
Multilevel configuration with material variants	CCUXMVAR	EXIT_SAPLCUKO_009	ZXCUCU20
Component quantity for set development	CCUXSETQ	EXIT_SAPLCUKO_010	ZXCUC2U01
Availability of customer functions in the configuration editor	CEI00000	EXIT_SAPLCEI0_010 through EXIT_SAPLCEI0_019	ZXCEI0U01 ZXCEI0U02 ZXCEI0U03
Configuration: Determine superior material	CUBX0001	EXIT_SAPLCUBX_001 EXIT_SAPLCUBX_002 EXIT_SAPLM60P_003	ZXCUBXU01 ZXCUBXU02 ZX60PU03
Additional logic deleting classification data from the LO-VC view	CUCPDELE	EXIT_SAPLCLDL_002 EXIT_SAPLCUCP_003 EXIT_SAPLCUCP_004 EXIT_SAPLCUCP_005	ZXCUCPU01 ZXCUCPU02 ZXCUCPU03 ZXCUCPU04
Additional logic deleting classification data from the LO-VC view CBASE	CUCPDEL1	EXIT_SAPLCUCP_006 EXIT_SAPLCUCP_007 EXIT_SAPLCUCP_008	ZXCUCPU05 ZXCUCPU06 ZXCUCPU07

The documentation on individual customer enhancements is in the SAP System with the enhancements themselves. To display the documentation on a customer enhancement:

1. Choose *Tools* → *ABAP Workbench* → *Utilities* → *Enhancements* → *Definition*.
2. Enter the technical name of the customer enhancement.
3. Select *Documentation* and choose  *Display*.

**See also:**

[Specifying Enhancements in the Configuration Editor \[Page 255\]](#)

## ALE Transfer of Configuration Data

### Use

This document describes ALE (Application Link Enabling) distribution of master data for variant configuration between R/3 systems. ALE can be used in R/3 Release 3.0, 3.1, and above.



For information on setting up ALE Customizing, see the SAP Library *Cross-Application Functions → (Business Framework Architecture) → ALE Business Process Library → ALE Quick Start*.

### Distribution of Master Data

This is the menu path for distributing master data:

up to 3.1: *Logistics → Central functions → Distribution*

as of 4.0: *Tools → (Business Framework) → ALE → Master Data Distribution*

or transaction BALE.

Examples of menu paths in the ALE transaction:

- Distribution of characteristics: *Logistics → Classification System → Characteristic → Send*
- Distribution of materials: *Cross-Application → Material → Send*

The data to be transferred is interdependent, so we advise you to transfer data in the sequence shown here.

However, your existing data may require you to use a different procedure or to transfer data multiple times.

#### 1. Characteristics and Characteristic Values

Most objects for variant configuration are dependent on characteristics. For this reason, characteristics must be transferred first.

*Characteristics with value hierarchies, long texts for characteristic values, or linked documents may lead to problems during transfer.*

Message type: CHRMAS

Availability: as of Release 3.0

#### 2. Classes

When you use ALE to distribute classes, the characteristic assignments are also transferred.

Message type: CLSMAS

Availability: as of Release 3.0

#### 3. Variant Table Structures

These are the variant tables that are created to support data maintenance.

Message type: VTAMAS

Availability: as of Release 3.1

#### 4. Variant Table Contents

Once the structures of the variant tables have been distributed, their contents can be transferred.

**ALE Transfer of Configuration Data**

Message type: VTMMAS

Availability: as of Release 3.1

**5. User-Defined Functions (Variant Functions, VC Functions)**

User-defined functions in variant configuration let you use function modules that you have written, to check and infer characteristic values.

The distribution of functions only transfers the framework (texts, characteristics, and so on).

The function modules that belong to the functions must be transferred first, using the usual R/3 transport system.

Message type: VFNMAS

Availability: as of Release 4.5

**6. Object Dependencies (Except Constraints)**

Dependencies (preconditions, selection conditions, procedures, and actions) usually refer to characteristics, characteristic values, variant tables, and variant functions. For this reason, dependencies must be distributed after this other master data. The dependencies transferred here are global dependencies. Local dependencies are transferred with the objects to which they are assigned. For example, if you created a selection condition as a local dependency for a BOM item, this dependency is transferred when you use ALE to distribute the BOM (bill of material).

Message type: KNOMAS

Availability: as of Release 3.1

**7. Constraints**

Constraints can only be distributed as of Release 4.5.

Message type: KNOMAS

Availability: as of Release 4.5

**8. Constraint Nets**

Constraint nets can only be distributed as of Release 4.5.

Message type: DEPNET

Availability: as of Release 4.5

**9. Assignment of Dependencies to Characteristics and Characteristic Values**

The characteristics are transferred once more to do this. Start ALE distribution for characteristics again, and the system transfers the assignments.

See point 1: Characteristics and Characteristic Values

**10. Configurable Materials**

Depending on how many configurable materials you have, you can use ALE to distribute them or create them in the target system. The other materials in BOMs for configurable materials are transferred to the target client by the usual data transfer.

*Material variants cannot be distributed by ALE.*

Message type: MATMAS

Availability: as of Release 3.0

**11. Configuration Profiles**

Configuration profiles must be distributed after the configurable materials, because the key of the material identifies the profile. During the ALE process, the dependencies for the



## ALE Transfer of Configuration Data

configuration profile are also transferred, and are reassigned to the profile in the target client.

Message type: CNPMAS  
Availability: as of Release 3.1

### 12. Classification

You only need to distribute classifications if you have classified materials.

Message type: CLFMAS  
Availability: as of Release 3.0

### 13. Bills of Material

Material BOMs for configurable materials can be distributed using ALE. The transfer also assigns the dependencies for BOM items to the items. Make sure that the dependencies and all the materials required already exist in the target system.

*If material variants are assigned to the BOM, there may be problems transferring the super BOM.*

Message type: BOMMAS  
Availability: as of Release 3.1

## Availability of Engineering Change Management for ALE Objects

Data created with engineering change management can be transferred as of the release shown below. However, first the change numbers must already exist in the target system. You can create them manually if required.

As of Release 3.0: material master records

As of Release 3.1: BOMs, dependencies, configuration profiles, and variant table contents

As of Release 4.0: characteristics, classes, and classifications

As of Release 4.5: constraints and constraint nets

## Message Types for Further Objects in the Variant Configuration Environment

Object	Message Type	Availability
Change objects	ECMMAS	as of Release 4.5
Conditions	COND_A	as of Release 3.1H
Cost centers	COSMAS	as of Release 3.0
Cost center groups	COGRP	as of Release 3.1
Activity type	COAMAS	as of Release 3.0
Activity type groups	COGRP5	as of Release 3.1
Cost elements	COELEM	as of Release 3.0

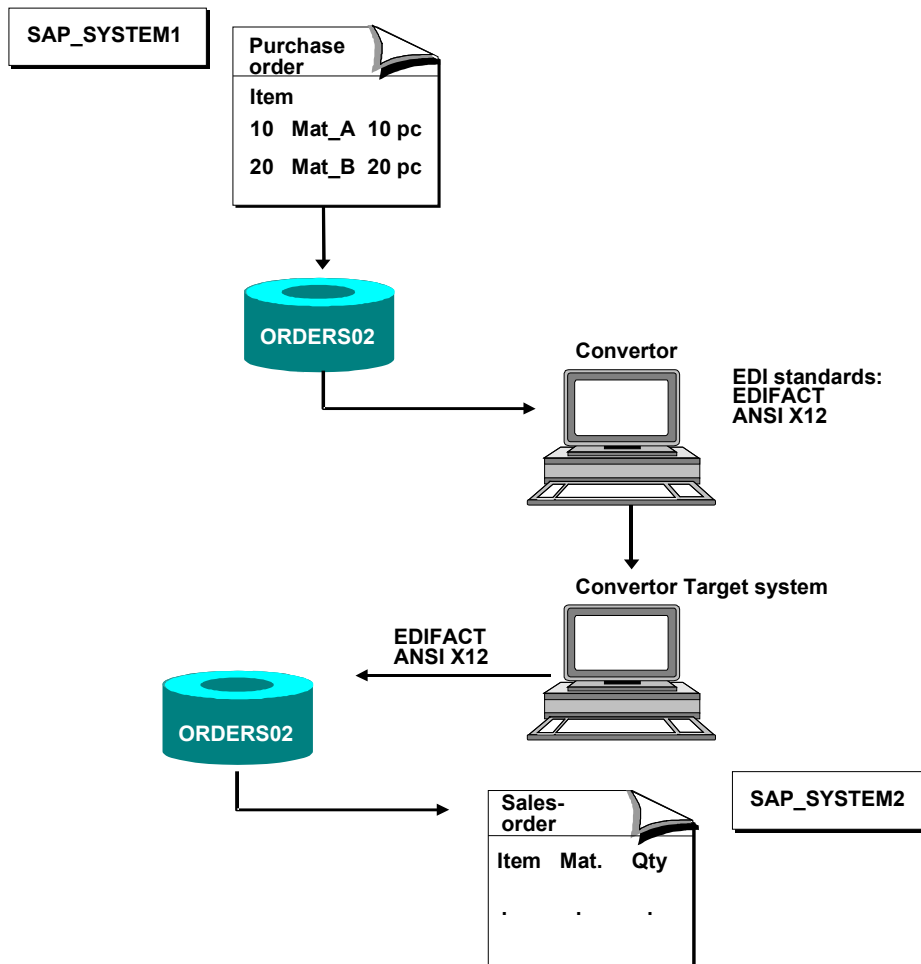
**ALE Transfer of Configuration Data**

Cost element groups	COGRP2	as of Release 3.1
Prices	COACTV	as of Release 3.0
Units of measure	COCOKA	as of Release 3.0

## EDI for KMATs (Information on Creating Your Own)

The following sections describe the structure of an intermediate document (IDoc) for configurable materials.

EDI (Electronic Data Interchange) enables you to electronically transfer documents, such as sales orders, quotations, purchase orders, from one R/3 System to another R/3 System.



For example, a purchase order is created for materials MAT\_A and MAT\_B in an R/3 System. You use EDI to transfer this purchase order to another R/3 System, such as the company from which the material is ordered. In the second R/3 System, the purchase order is converted to a sales order. This assumes that the master data in the source system is also maintained in the target system.

The application data is stored in an intermediate document (IDoc). Each IDoc is assigned to a specific IDoc type, which defines the structure and format of the electronically transferred data. If a document is to contain configurable objects, you must use IDoc type ORDERS02.

An IDoc comprises various segments, each of which carries specific information. For example, one segment can contain the header data of a document and another can contain the

**EDI for KMATs (Information on Creating Your Own)**

organizational units. Each segment has a segment type, which defines the type of data that a segment can contain.



For more general information on EDI processing, see the R/3 Library, under *CA EDI and the IDoc Interface*.

<b>Documents with Configurable Materials:</b>	<b>Scope of Configuration:</b>
<ul style="list-style-type: none"><li>• Create order</li></ul>	<ul style="list-style-type: none"><li>• Single-level/multi-level</li></ul>
<ul style="list-style-type: none"><li>• Create order with reference to a quotation</li></ul>	<ul style="list-style-type: none"><li>• Single-level/multi-level</li></ul>
<ul style="list-style-type: none"><li>• Confirm order</li></ul>	<ul style="list-style-type: none"><li>• Single-level/multi-level</li></ul>
<ul style="list-style-type: none"><li>• Change order</li></ul>	<ul style="list-style-type: none"><li>• Single-level/multi-level</li></ul>
<ul style="list-style-type: none"><li>• Create inquiry</li></ul>	<ul style="list-style-type: none"><li>• Single-level/multi-level</li></ul>
<ul style="list-style-type: none"><li>• Purchase order</li></ul>	<ul style="list-style-type: none"><li>• Single-Level</li></ul>

**IDoc Procedure for a Configurable Material (Inbound IDoc):**

Once an IDoc is received, the data is transferred to the screen fields of the transaction in batch mode using Call Transaction calls.

However, this procedure is not possible for the configuration data of an item, because the screen sequence for a configuration is very variable. For this reason, the configuration data is first put into memory. If it is then recognized that a document item in the sales order data is a configurable material, function modules for loading the data from memory and transferring it to the configurator are called.

## Basic Type ORDERS02

This section describes the structure of basic type ORDERS02, which is defined for purchasing and sales documents with configurable objects.



The configuration data is shown in the segment types in bold print.

### Basic Type ORDERS02:

<b>ORDERS02</b>	<b>Purchasing/Sales</b>
E1EDK01	IDoc: document header general data
E1EDK14	IDoc: document header organizational data
E1EDK03	IDoc: document header date segment
E1EDK04	IDoc: document header taxes
E1EDK05	IDoc: document header conditions
E1EDKA1	IDoc: document header partner information
E1EDKA3	IDoc: document header partner information additional data
E1EDK02	IDoc: document header reference data
E1EDK17	IDoc: document header terms of delivery
E1EDK18	IDoc: document header terms of payment
E1EDKT1	IDoc: document header text identification
E1EDKT2	IDoc: document header texts
E1EDP01	IDoc: document item general data
E1EDP02	IDoc: document item reference data
<b>E1CUREF</b>	CU: reference order item / instance in configuration
E1EDP03	IDoc: document item date segment
E1EDP04	IDoc: document item taxes
E1EDP05	IDoc: document item conditions
E1EDP20	IDoc: schedule lines
E1EDPA1	IDoc: document item partner information
E1EDPA3	IDoc: document item partner information additional data
E1EDP19	IDoc: document item object identification
E1EDP17	IDoc: document item terms of delivery
E1EDP18	IDoc: document item terms of payment
E1EDPT1	IDoc: document item text identification

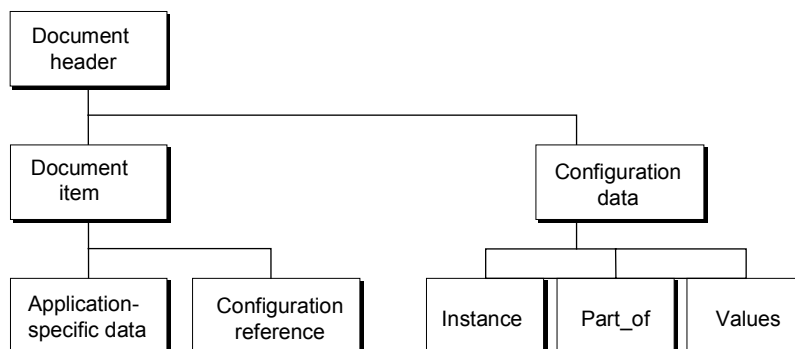
**Basic Type ORDERS02**

E1EDPT2	IDoc: document item texts
E1EDC01	LV: service specifications general data
E1EDC02	Service specifications item: reference data
E1EDC03	Service specifications item: date segment
E1EDC04	Service specifications item: taxes
E1EDC05	Service specifications item: conditions
E1EDCA1	Service specifications item: partner information
E1EDC19	Service specifications item: object identification
E1EDC17	Service specifications item: terms of delivery
E1EDC18	Service specifications item: terms of payment
E1EDCT1	Service specifications item: text identification
E1EDCT2	Service specifications item: texts
<b>E1CUCFG</b>	CU: configuration data
<b>E1CUINS</b>	CU: instance data
<b>E1CUPRT</b>	CU: part_of data
<b>E1CUVAL</b>	CU: assigned values
E1EDS01	IDoc: totals general segment

The data of the document item and the configuration data are stored separately in an IDoc. This makes it clear where the configuration data starts and ends. Segment type E1EDP01 contains the general document data of an order item. Segment type E1CUCFG contains the configuration data that is on the same level as the document data.

The connection between the document item and the configuration data is established by segment type E1CUREF.

Segment E1CUCFG identifies the general configuration data and contains the instances (configurable objects), the relationships between the instances (superior/subordinate object), and the characteristic values assigned to individual instances in segment types E1CUINS, E1CUPRT, and E1CUVAL.





---

**Segment Type E1CUREF**

## Segment Type E1CUREF

This segment type is used to establish the connection between the SD data of a document item and the configuration data.

Segment E1CUREF is a sub-segment of segment E1EDP01, which contains the general item data such as quantity, unit of measure, item category, and so on.

Since configuration data is stored separately from the document item data, this segment is the interface between the two segments. It ensures that the correct configuration data is assigned to an item.

This segment contains the following fields:

### POSEX

The external item number is used to identify the configuration data for a document item. The external item number is copied from the purchase order to the *Purchase order item* field in the purchasing data in the sales order. This means that the sales order item is assigned to exactly one purchase order item.

The value in this field must match the value in field POSEX in segment E1EDP01.

### CONFIG\_ID

The configuration data is identified by the CONFIG\_ID.

### INST\_ID

The configurable material of the document item is identified by the INST\_ID.



## Segment Type E1CUCFG

### Use

This segment type is used to assign the configuration data to a document item.

### Features

This segment contains the following fields:

#### POSEX

The POSEX field identifies the document item to which the configuration data is assigned. This item number is copied from the purchase order to the *Purchase order item* field in the order.

#### CONFIG\_ID

The configuration data is identified by the CONFIG\_ID. You can assign any CONFIG\_ID you want, but the CONFIG\_ID must be unique within an IDoc so that you can transfer the configuration of different order items and keep the data separate.



Item 10	BOX	5 pc	CONFIG_ID: 000010
Item 20	BOX	7 pc	CONFIG_ID: 000020

#### ROOT\_ID

The instance number (INST\_ID) entered in the ROOT\_ID field must be the instance number of the top-level item in the configuration. It describes the starting point in the hierarchy for the configuration process. This top-level instance matches the document item in segment E1EDP01.

The instance number is used to identify the materials in the IDoc.

#### COMPLETE and CONSISTENT

These two fields give the status of the configuration, using T = true and F = false. COMPLETE defines whether the configuration is complete, and CONSISTENT defines whether the configuration is consistent.

**Other field names:** fields SCE, KBNAME, KBVERSION, CFGINFO, KBPROFILE, KBLANGUAGE, CBASE\_ID, and CBASE\_ID\_TYPE are filled automatically. They are used internally for communication between R/3 and other mySAP.com components.

**Segment Type E1CUINS**

## Segment Type E1CUINS

### Use

In this segment type, the individual configurable materials in a configuration are described. A configuration can have any number of segments with instance data.

### Features

This segment contains the following fields:

#### INST\_ID

In an IDoc, each configurable material has an INST\_ID that uniquely identifies the data of an object. For an inbound IDoc, the instance number is assigned externally, and must be unique within a configuration. For an outgoing IDoc, the numbers are assigned by SAP.



This instance number is completely separate from any internal instance number of the configurator.

#### OBJ\_TYPE:

Under OBJ\_TYPE, the object type of an instance is entered (for example, MARA for materials).

#### CLASS\_TYPE

The instance in a configuration can also be a class. In this case, the class type is entered instead of the object type.



Transferred instances cannot be classes. Only object types can be transferred.

#### OBJ\_KEY

OBJ\_KEY is the object key (for example, the material number or class name).



The object key must be transferred in upper-case letters, because the data is not automatically converted to upper-case letters.

#### OBJ\_TXT

Here, the language-dependent description of an object is entered. This field is only filled out for outgoing IDocs. Otherwise, this field is not relevant.

#### QUANTITY

This is the quantity in which the object is part of the configuration. This is the component quantity from the BOM or the instance quantity, not the cumulative quantity from the sales order.

#### QUANTITY\_UNIT

This is the unit of measure for the quantity – for example, pc = piece.

#### COMPLETE and CONSISTENT

**Segment Type E1CUINS**

These two fields give the status of the configuration at instance level. Use the entries T = True and F = False. COMPLETE defines whether the instance is complete, and CONSISTENT defines whether the instance is consistent.

**Other field names:** fields AUTHOR, OBJECT\_GUID, PERSIST\_ID, and PERSIST\_ID\_TYPE are filled automatically. They are used internally for communication between R/3 and other mySAP.com components.

**Segment Type E1CUPRT**

## Segment Type E1CUPRT

### Use

In this segment type, the relationships between the individual configurable materials are defined. These relationships define how the instances are linked together in the hierarchy.

As well as the data on the hierarchical relationships, this segment also determines which master data object in the decomposition (BOM) an instance comes from. It is important to describe this uniquely, because this determines which BOM item fields are included from which BOM item. For example, you can define that an item is relevant to sales and appears in the sales order as a sub-item of the main item.

### Features

This segment contains the following fields:

PARENT\_ID

Identifier of the superior object (in the hierarchy) of the object entered under INST\_ID.

INST\_ID

Identifier of the object that is subordinate to the instance entered under PARENT\_ID.

PART\_OF\_NO

To ensure that the value data is linked to the correct master data object, the BOM item number is also transferred.

If several items in the BOM have the same item number, and also have the same component data (for example, material number, or class type and class), the first item in the BOM is always assumed.



To ensure that the data is entered in the correct BOM item, the system searches for the item using first the item number, then the object type, then the object key or class type. If no BOM item is found for these criteria, the system searches again using just the object type, then the object key or class type.

OBJ\_TYPE

Object type of the master data object from which the instance under INST\_ID originates (for example, MARA for material or KLAH for class).

CLASS\_TYPE

The class type is part of the external key of a BOM item that is a class item.



Transferred instances cannot be classes. Only object types can be transferred.

OBJ\_KEY

Object key, such as material number or class name.



The object key must be transferred in upper-case letters, because the data is not automatically converted to upper-case letters.

**Other field names:** fields AUTHOR, SALES\_RELEVANT, and PART\_OF\_GUID are filled automatically. They are used internally for communication between R/3 and other mySAP.com components.

**Segment Type E1CUVAL**

## Segment Type E1CUVAL

### Use

Segment type E1CUVAL contains the data on assigned values. Each characteristic has a separate segment.



The characteristic name (field CHARC) must be transferred in upper-case letters, because the data is not automatically converted to upper-case letters

The settings in the characteristic determine whether the characteristic values are case sensitive. If the values of the characteristic are defined as case sensitive, values that are transferred in upper-case letters may not be recognized.

### Features

This segment contains the following fields:

INST\_ID

The INST\_ID shows which configurable material the values are assigned to.

CHARC

Characteristic name

CHARC\_TXT

Language-dependent description of the characteristic

VALUE

Lowest characteristic value of the allowed values

VALUE\_TXT

Language-dependent description of the characteristic value

VALUE\_TO

Highest characteristic value of the allowed values

VALCODE

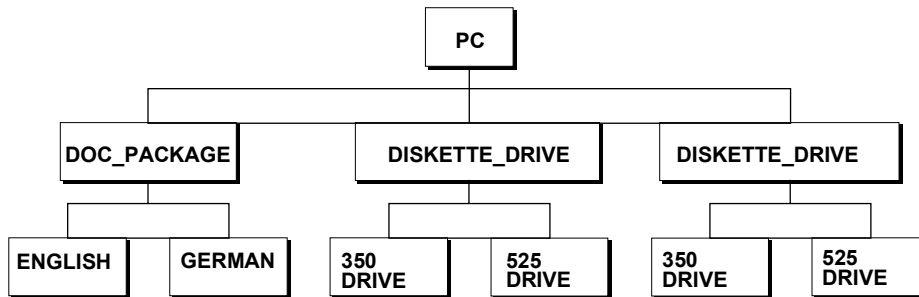
Shows the relationship between the lowest allowed value and the highest allowed value (for example, 1-3 or -1-3). The fixed values are on the *Allowed values* tab of domain CUX\_VALCOD.

If the VALCODE field is not filled, the fixed value 1 automatically applies. This means that the lower limit of the allowed values applies.

The AUTHOR field is filled automatically. It is used internally for communication between R/3 and other mySAP.com components.

## Creating an Order for Configurable Materials with EDI

The following example shows the IDoc structure for a PC. In the configuration profile, the BOM explosion result is defined as *Sales order*, so the items that are relevant to sales can be configured individually and appear in the sales order as sub-items of the main item. The PC consists of other configurable materials: one or two diskette drives, and two packages with country-specific documentation.



The example uses a sales order with 2 items, which are configured differently.

Item	Matl #	Qty
10	PC	7
20	PC	11

The items are configured as follows:

Item	Instance	Values		
10	PC	DISKETTE_DRIVE_NUMBER	=	2
	DISKETTE_DRIVE	DISKETTE_DRIVE_TYPE	=	3,5
	DISKETTE_DRIVE	DISKETTE_DRIVE_TYPE	=	5,25
	DOC_PACKAGE	Language	=	English
20	PC	DISKETTE_DRIVE_NUMBER	=	1
	DISKETTE_DRIVE	DISKETTE_DRIVE_TYPE	=	5,25
	DOC_PACKAGE	Language	=	German

The following components are copied to the sales order as items, according to the configuration:

Item	Main item	Material	Qty
10	-	PC	7
20	10	Diskette_drive	7
30	10	Diskette_drive	7
40	10	DOC_PACKAGE	14
50	-	PC	11

**Creating an Order for Configurable Materials with EDI**

60	50	Diskette_drive	11
70	50	DOC_PACKAGE	22

The IDoc for this sales order has the following structure:

[Example: IDoc Structure \[Page 305\]](#)

The connection between sales order item and configuration data is defined in segment E1CUREF.

[Example: Segment E1CUREF \[Page 307\]](#)

The general configuration data of the sales order item is described in segment E1CUCFG:

[Example: Segment E1CUCFG \[Page 308\]](#)

The individual instances in the configuration are described in segment E1CUINS:

[Example: Segment E1CUINS \[Page 309\]](#)

The relationships between the individual instances are defined in segment E1CUPRT:

[Example: Segment E1CUPRT \[Page 312\]](#)

The values assigned to the individual instances are entered in segment E1CUVAL:

[Example: Segment E1CUVAL \[Page 314\]](#)



## IDoc Structure

The IDoc comprises 41 datasets. For sales order items 10 and 20, segments 14 and 17 were created with segment type E1EDP01 to contain general item data. A segment of type E1CUREF, referring to the configuration data of the item, exists for each order item. The configuration data of the items is in the E1CUCFG segments and sub-segments.



The sequence in which segment types are listed in the IDoc must match the basic structure of the IDoc.

IDoc number      **755252**

IDoc type         **ORDERS02**

Number	Segment	Description
1	E1EDK01	IDoc: general document header data
2	E1EDK14	IDoc: document header organizational data
3	E1EDK14	IDoc: document header organizational data
4	E1EDK14	IDoc: document header organizational data
5	E1EDK14	IDoc: document header organizational data
6	E1EDK03	IDoc: document header date segment
7	E1EDK03	IDoc: document header date segment
8	E1EDK03	IDoc: document header date segment
9	E1EDKA1	IDoc: document header partner data
10	E1EDKA1	IDoc: document header partner data
11	E1EDKA1	IDoc: document header partner data
12	E1EDK02	IDoc: document header reference data
13	E1EDK17	IDoc: document header terms of delivery
14	E1EDP01	IDoc: document item general data
<b>15</b>	<b>E1CUREF</b>	<b>CU: reference order item / instance in configuration</b>
16	E1EDP19	IDoc: document item object identifier
17	E1EDP01	IDoc: document item general data
<b>18</b>	<b>E1CUREF</b>	<b>CU: reference order item / instance in configuration</b>
19	E1EDP19	IDoc: document item object identifier
<b>20</b>	<b>E1CUCFG</b>	<b>CU: configuration data</b>

## IDoc Structure

21	E1CUINS	CU: instance data
22	E1CUINS	CU: instance data
23	E1CUINS	CU: instance data
24	E1CUINS	CU: instance data
25	E1CUPRT	CU: part_of data
26	E1CUPRT	CU: part_of data
27	E1CUPRT	CU: part_of data
28	E1CUVAL	CU: characteristic values
29	E1CUVAL	CU: characteristic values
30	E1CUVAL	CU: characteristic values
31	E1CUVAL	CU: characteristic values
32	E1CUCFG	CU: configuration data
33	E1CUINS	CU: instance data
34	E1CUINS	CU: instance data
35	E1CUINS	CU: instance data
36	E1CUPRT	CU: part_of data
37	E1CUPRT	CU: part_of data
38	E1CUVAL	CU: characteristic values
39	E1CUVAL	CU: characteristic values
40	E1CUVAL	CU: characteristic values
41	E1EDS01	IDoc: sum segment general

## Example: Segment E1CUREF

Field	DE Structure	Value
POSEX	CHAR6	000010
CONFIG_ID	CHAR6	000010
INST_ID	CHAR8	00000001
POSEX	CHAR6	000020
CONFIG_ID	CHAR6	000020
INST_ID	CHAR8	00000005

This segment links the relevant configuration data to the 2 order items. The items are identified by the external item number (POSEX). The PC in order item 10 has external item number 000010. The PC in order item 20 has external item number 000020. In segment E1EDP19, the same external item number must be used.

The INST\_ID identifies the material in the order item. The identifier 00000001 refers to the PC in document item 10. The identifier 00000005 refers to the PC in document item 20. The CONFIG\_ID identifies the configuration data for the materials.

**Example: Segment E1CUCFG****Example: Segment E1CUCFG**

Field	DE Structure	Value
POSEX	CHAR6	000010
CONFIG_ID	CHAR6	000010
ROOT_ID	CHAR8	00000001
COMPLETE	CU_CHECKED	T
CONSISTENT	CU_CHECKED	T
POSEX	CHAR6	000020
CONFIG_ID	CHAR6	000020
ROOT_ID	CHAR8	00000005
COMPLETE	CU_CHECKED	F
CONSISTENT	CU_CHECKED	F

Segment E1CUCFG contains the general configuration data for the configurable material in the order item.

Each E1CUCFG segment must be accessible from exactly one E1CUREF segment. Otherwise, the configuration data is ignored. In other words, POSEX and CONFIG\_ID in an E1CUREF segment must point to a suitable E1CUCFG segment. The POSEX and CONFIG\_ID fields in this segment then have the same values as in the E1CUREF segment.

Below the E1CUCFG segments are the datasets for the individual instances in the configuration (E1CUINS), the relationships of these instances to each other (E1CUPRT), and the characteristic values assigned to them (E1CUVAL).

The configuration with CONFIG\_ID 10 is complete and consistent. The configuration with CONFIG\_ID 20 is incomplete and inconsistent. The status fields COMPLETE and CONSISTENT can be filled, but need not be. If they are not filled, the configurator infers these values automatically. If you send field value "T", but the configurator finds an inconsistency, the configurator overrides the IDoc entry. Status "F" is not overridden by the configurator, because it is interpreted as a user entry.

## Example: Segment E1CUINS

This segment describes all the configurable materials (instances) for which configuration data was transferred.

Instance 00000001 identifies the PC in order item 10, of which 7 were ordered. Two diskette drives were selected for the PC: instance 00000002 identifies the first diskette drive of the PC and instance number 00000003 identifies the second diskette drive of the PC. Instance 00000004 identifies the documentation package, of which 2 are supplied with the PC. The quantity for the instances that are part of the PC is the BOM quantity (instance quantity), not the cumulative quantity from the sales order.

Instance 00000005 identifies the PC in order item 20, of which 11 were ordered. One diskette drive was selected for the PC. Instance 00000006 identifies this diskette drive. Instance 00000007 identifies the documentation package that was selected for the PC, of which 2 were ordered.

Almost every instance on the configuration level is complete and consistent (T = True; F = False). The status fields can be filled, but need not be. If they are not filled, the configurator infers these values automatically.

Field	DE Structure	Value
INST_ID	CHAR	00000001
OBJ_TYPE	CHAR10	MARA
CLASS_TYPE	CHAR3	
OBJ_KEY	CHAR50	PC
OBJ_TXT	CHAR70	
QUANTITY	CHAR15	7
QUANTITY_UNIT	CUX_QUAN_UNIT	pc
COMPLETE	CU_CHECKED	T
CONSISTENT	CU_CHECKED	T
INST_ID	CHAR	00000002
OBJ_TYPE	CHAR10	MARA
CLASS_TYPE	CHAR3	
OBJ_KEY	CHAR50	DISKETTE_DRIVE
OBJ_TXT	CHAR70	
QUANTITY	CHAR15	1
QUANTITY_UNIT	CUX_QUAN_UNIT	pc
COMPLETE	CU_CHECKED	T
CONSISTENT	CU_CHECKED	T
INST_ID	CHAR	00000003
OBJ_TYPE	CHAR10	MARA

**Example: Segment E1CUINS**

CLASS_TYPE	CHAR3	
OBJ_KEY	CHAR50	DISKETTE_DRIVE
OBJ_TXT	CHAR70	
QUANTITY	CHAR15	1
QUANTITY_UNIT	CUX_QUAN_UNIT	pc
COMPLETE	CU_CHECKED	T
CONSISTENT	CU_CHECKED	T
INST_ID	CHAR	00000004
OBJ_TYPE	CHAR10	MARA
CLASS_TYPE	CHAR3	
OBJ_KEY	CHAR50	DOC_PACKAGE
OBJ_TXT	CHAR70	
QUANTITY	CHAR15	2
QUANTITY_UNIT	CUX_QUAN_UNIT	pc
COMPLETE	CU_CHECKED	F
CONSISTENT	CU_CHECKED	F
INST_ID	CHAR	00000005
OBJ_TYPE	CHAR10	MARA
CLASS_TYPE	CHAR3	
OBJ_KEY	CHAR50	PC
OBJ_TXT	CHAR70	
QUANTITY	CHAR15	11
QUANTITY_UNIT	CUX_QUAN_UNIT	pc
COMPLETE	CU_CHECKED	T
CONSISTENT	CU_CHECKED	T
INST_ID	CHAR	00000006
OBJ_TYPE	CHAR10	MARA
CLASS_TYPE	CHAR3	
OBJ_KEY	CHAR50	DISKETTE_DRIVE
OBJ_TXT	CHAR70	
QUANTITY	CHAR15	1
QUANTITY_UNIT	CUX_QUAN_UNIT	pc
COMPLETE	CU_CHECKED	T

Example: Segment E1CUINS

CONSISTENT	CU_CHECKED	T
INST_ID	CHAR	00000007
OBJ_TYPE	CHAR10	MARA
CLASS_TYPE	CHAR3	
OBJ_KEY	CHAR50	DOC_PACKAGE
OBJ_TXT	CHAR70	
QUANTITY	CHAR15	2
QUANTITY_UNIT	CUX_QUAN_UNIT	pc
COMPLETE	CU_CHECKED	F
CONSISTENT	CU_CHECKED	F

**Example: Segment E1CUPRT****Example: Segment E1CUPRT**

This segment contains the BOM structure for the configurable materials. The materials are identified by their instance numbers. This segment also contains the key fields of the BOM item for the components. This ensures that the correct component is read if several BOM items contain the same material.

The PC in order item 10 (instance 00000001) has two diskette drives (instances 00000002 and 00000003) and one documentation package (instance 00000004). The PC in order item 20 (instance 00000005) has one diskette drive (instance 00000006) and one documentation package (instance 00000007). The instances are described in the E1CUINS segments.

Field	DE Structure	Value
PARENT_ID	CHAR8	00000001
INST_ID	CHAR8	00000002
PART_OF_NO	CHAR4	0020
OBJ_TYPE	CHAR10	MARA
CLASS_TYPE	CHAR3	
OBJ_KEY	CHAR50	DISKETTE_DRIVE
PARENT_ID	CHAR8	00000001
INST_ID	CHAR8	00000003
PART_OF_NO	CHAR4	0030
OBJ_TYPE	CHAR10	MARA
CLASS_TYPE	CHAR3	
OBJ_KEY	CHAR50	DISKETTE_DRIVE
PARENT_ID	CHAR8	00000001
INST_ID	CHAR8	00000004
PART_OF_NO	CHAR4	0040
OBJ_TYPE	CHAR10	MARA
CLASS_TYPE	CHAR3	
OBJ_KEY	CHAR50	DOC_PACKAGE
PARENT_ID	CHAR8	00000005
INST_ID	CHAR8	00000006
PART_OF_NO	CHAR4	0020
OBJ_TYPE	CHAR10	MARA
CLASS_TYPE	CHAR3	
OBJ_KEY	CHAR50	DISKETTE_DRIVE
PARENT_ID	CHAR8	00000005



INST_ID	CHAR8	00000007
PART_OF_NO	CHAR4	0040
OBJ_TYPE	CHAR10	MARA
CLASS_TYPE	CHAR3	
OBJ_KEY	CHAR50	DOC_PACKAGE

**Example: Segment E1CUVAL****Example: Segment E1CUVAL**

This segment contains the characteristic values for a configurable material. The materials are identified by their instance numbers.

The PC in order item 10 is to include 2 diskette drives: one 3.5" drive and one 5.25" drive. For this reason, the value '2' was assigned to characteristic DISKETTE\_DRIVE\_NUMBER. Characteristic DISKETTE\_DRIVE\_TYPE has the value '3.5' for the first diskette drive and '5.25' for the second diskette drive.

The documentation for the PC is to be supplied in English. For this reason, characteristic DOC\_LANGUAGE has the value 'English' for material DOC\_PACKAGE.

The PC in order item 20 is only to have one (5.25") diskette drive. The documentation is to be supplied in German. For this reason, the value '1' was assigned to characteristic DISKETTE\_DRIVE\_NUMBER. Characteristic DISKETTE\_DRIVE\_TYPE has the value '5.25' for material DISKETTE\_DRIVE. Characteristic DOC\_LANGUAGE has the value 'German' for material DOC\_PACKAGE.

Field	DE Structure	Value
INST_ID	CHAR8	00000001
CHARC	CHAR40	DISKETTE_DRIVE_NUMBER
CHARC_TXT	CHAR70	
VALUE	CHAR40	2
VALUE_TXT	CHAR70	
VALUE_TO	CU_VALUE	
VALCODE	CUX_VALCOD	
INST_ID	CHAR8	00000002
CHARC	CHAR40	DISKETTE_DRIVE_TYPE
CHARC_TXT	CHAR70	
VALUE	CHAR40	3.5
VALUE_TXT	CHAR70	
VALUE_TO	CU_VALUE	
VALCODE	CUX_VALCOD	
INST_ID	CHAR8	00000003
CHARC	CHAR40	DISKETTE_DRIVE_TYPE
CHARC_TXT	CHAR70	
VALUE	CHAR40	5.25
VALUE_TXT	CHAR70	
VALUE_TO	CU_VALUE	
VALCODE	CUX_VALCOD	

Example: Segment E1CUVAL

INST_ID	CHAR8	00000004
CHARC	CHAR40	DOC_LANGUAGE
CHARC_TXT	CHAR70	
VALUE	CHAR40	English
VALUE_TXT	CHAR70	
VALUE_TO	CU_VALUE	
VALCODE	CUX_VALCOD	
INST_ID	CHAR8	00000005
CHARC	CHAR40	DISKETTE_DRIVE_NUMBER
CHARC_TXT	CHAR70	
VALUE	CHAR40	1
VALUE_TXT	CHAR70	
VALUE_TO	CU_VALUE	
VALCODE	CUX_VALCOD	
INST_ID	CHAR8	00000006
CHARC	CHAR40	DISKETTE_DRIVE_TYPE
CHARC_TXT	CHAR70	
VALUE	CHAR40	5.25
VALUE_TXT	CHAR70	
VALUE_TO	CU_VALUE	
VALCODE	CUX_VALCOD	
INST_ID	CHAR8	00000007
CHARC	CHAR40	DOC_LANGUAGE
CHARC_TXT	CHAR70	
VALUE	CHAR40	German
VALUE_TXT	CHAR70	
VALUE_TO	CU_VALUE	
VALCODE	CUX_VALCOD	

## Error Handling

### Error Handling

If you want to be able to trace the configuration process, you need to activate the trace function. You see the dialog box for activating the trace function when you process a configurable item, before you start the configuration (see [Trace Function \[Page 244\]](#)). However, this only applies if you run an IDoc in the foreground.

If errors occur, you see a trace message, even if you did not activate the trace function.

If processing was terminated due to errors, you see error messages that attempt to tell you why.

Here are some possible causes of error:

- An incorrect characteristic name was transferred.
- An incorrect characteristic value was transferred.
- The characteristic value transferred does not match the format of the characteristic.
- No value was transferred for a characteristic.

These errors may trigger other errors, so that part of the configuration cannot be processed.



Processing is terminated because an incorrect characteristic name was transferred. This may mean that selection conditions are not fulfilled, so that objects for which configuration data is transferred are not included in the configuration. You see a trace message, telling you that instances in the configuration could not be processed. If you are processing the configuration in the background, processing is terminated. If you are processing the configuration in the foreground, you see a list of the input data that could not be processed. The program then assumes that you know about this data, and does not terminate.

You also see the configuration structure dialog box, which allows you to configure the configurable materials that were not processed.

If an incorrect value was entered for a characteristic, you see a dialog box containing the allowed values. You can then select one of these values for the characteristic.

## Restrictions

The following restrictions apply to IDoc processing for configuration:

- (Inbound IDoc)
 

SD-specific data cannot be transferred for subordinate document items.

For example, one red bag and one green bag are selected as components of a material in the configuration of the material. You cannot transfer SD data (for example, that the red bag is supplied by vendor X and the green bag is supplied by vendor Y) in the IDoc for these two components.
- (Inbound IDoc)
 

The instances transferred cannot be classes. Only object types (for example, MARA) can be transferred.
- (Inbound IDoc)
 

The user cannot set quantities. Only the quantities that are inferred by dependencies or maintained in the master data of the decomposition item are valid.
- (Inbound IDoc)
 

You cannot select parts manually. A component is only included in the configuration if it is a non-variable component (used in all variants of the material) or if the selection condition for the component is fulfilled.
- (Inbound IDoc)
 

In certain situations, the load function may not be able to process input data for an instance, due to the processing sequence.

For example, the BOM of the header material contains components 1 and 2, which are configurable. When the header material is configured, component 1 is not selected. Component 2 is selected. The configuration of component 2 triggers a dependency that leads to component 1 being selected after all. However, the system may no longer recognize that data for component 1 is available. In these cases, the data is marked as not processed.
- (Outgoing IDoc)
 

The order confirmation always uses the data in the sales order to fill out the configuration data in the segments.

---

Creating a Knowledge Base Object for the SCE

## Creating a Knowledge Base Object for the SCE

### Use

The configurator in the R/3 System requires a range of objects for configuration (classes, characteristics, characteristic values, materials, bills of material (BOMs), dependencies, and so on). The objects required to configure a material are known as the knowledge base.

To use the Sales Configuration Engine (SCE), you need to extract this knowledge base from the R/3 System and export it to a PC.

To identify a knowledge base, you maintain a knowledge base object, which acts as parentheses around the knowledge base for a configurable material. This knowledge base object lets you collect all the knowledge belonging to the knowledge base.

The following objects make up the knowledge base:

- Materials
- Characteristics and their values
- Dependencies
- Configuration profiles
- Classes
- BOMs
- Variant tables
- Interfaces to user-defined functions



Task lists (for example, routings) are not loaded.

### Features

A knowledge base object contains one or more profiles that refer to a configurable material or a class of configurable materials. The configurable material or class is entered in the knowledge base profile as an [OO Class \[Page 332\]](#) (class in the general sense of a system of objects).

The knowledge base resulting from the knowledge base object lets you configure all configurable materials that are either entered in the profile or allocated to a class that is entered in the profile. You always start configuration with the OO class entered in the profile.



- a. A class has the configurable materials DESKTOP, LAPTOP, and WORKSTATION. If you enter a class in the knowledge base profile, the knowledge base contains all three materials. However, configuration in the SCE always starts with the class, from which you can then select the LAPTOP, for example.
- b. If you want to start configuration with the LAPTOP, create a second profile for the knowledge base object and enter configurable material LAPTOP in this profile.

---

**Creating a Knowledge Base Object for the SCE**

The second profile does not increase the size of the resulting knowledge base, because the LAPTOP is already loaded to the knowledge base by the first profile and is not loaded again.

**Activities**

1. Create a knowledge base object. The knowledge base object contains the knowledge base profiles for collecting objects for the configuration model.
2. Generate a runtime version of the knowledge base for the knowledge base object. The runtime version contains all objects that are valid on the date entered.
3. Export the runtime version to your PC.

## Creating a Knowledge Base Object

## Creating a Knowledge Base Object

### Procedure

1. From the variant configuration menu, choose *Knowledge base* → *Knowledge base object* → *Create*.

Enter a name for your knowledge base object.

Confirm your entry.

2. You see the basic data screen.

- Enter language-dependent descriptions for your knowledge base object.
- Enter a status for the knowledge base object.

You cannot create a runtime version of a knowledge base object that has the status *In preparation* or *Locked*.

- You can assign a group to the knowledge base object, to help you find the knowledge base object.

3. Choose *Goto* → *Profiles*.

The knowledge base profile identifies a configurable material or a class of configurable materials, and defines the user interface for configuration.

If you enter more than one profile for a knowledge base object, you see the profiles when you load the knowledge base to the SCE, and you can select a profile. As well as determining the start point for configuration, the knowledge base profile determines which user interface is started, because you can enter a user-specific user interface in the profile.



You can create profiles with different user interfaces for a configurable material. This does not duplicate the objects in the knowledge base.

4. Enter a name and description for the knowledge base profile.

To enter names for the profile in different languages, choose *Descriptions*.

Confirm.

5. Enter the OO class (object-oriented class), which is a configurable material or a class of configurable materials.

Enter the class type of the class. You must also enter a class type for a configurable material, so that the classes and characteristics for the material can be loaded.



If you enter a class type that is not in the configuration profile, the class type in the configuration profile overrides the class type you enter.

You also define which user interface is used for configuring the material:

- If you do not enter a user interface, the standard user interface of the R/3 System is loaded to the SCE.



### Creating a Knowledge Base Object

- If you have defined your own user interface, enter the name in the *UINAME* field. This name specifies the JAVA class in which the interface is implemented.



You cannot maintain tasks and events for a knowledge base object.

6. Save the knowledge base object.

### Result

The knowledge base object has been created. Now you can create a runtime version of the knowledge base.

## Creating a Runtime Version

## Creating a Runtime Version

### Prerequisites

You have created a knowledge base object as a formal parentheses around the objects of a knowledge base. When you create a runtime version, you collect all the objects that are in the knowledge base on a certain date.

### Procedure

1. Choose *Knowledge base* → *Runtime version* → *Create*. Enter the knowledge base object for which you want to create a runtime version, and enter a name for the version.

2. On the next screen, enter the following data:

- The date for creating the runtime version
- The status of the runtime version

You cannot download runtime versions with the status *In preparation* or *Locked to a PC*.

- The language for language-dependent descriptions and documentation for objects (such as characteristics and dependencies) for the runtime version

If you do not enter a language, the descriptions and documentation are loaded in all languages.



We advise you to load descriptions in all languages, so that missing language-dependent descriptions do not cause errors.

- You can only create a version for one plant and one BOM application. For this reason, enter the plant and BOM application. The runtime version only reads the BOMs that are selected by the application.
- If you want to load actions for the configuration profile, set the *Incl. actions* indicator. These actions are automatically converted to procedures, so that you can define the sequence in which they are processed.



Actions and procedures for characteristics and characteristic values are never loaded. If you want to load these actions and procedures, reallocate them to the configuration profile.

3. Generate the runtime version. This collects the objects described under *Objects of a runtime version*.

If errors occur when generating the runtime version, you see an error message and the process is terminated.

## Changing a Runtime Version

### Prerequisites

You have created a runtime version for a knowledge base object. Objects in the knowledge base have been changed since then. To show these changes in the runtime version, you must regenerate the runtime version.

### Procedure

1. Choose *Knowledge base* → *Runtime version* → *Change*.
2. Check the date for the runtime version – the changes must be valid on the date you enter. When you change a runtime version, the default date is the date when the runtime version was created, not today's date. Change the date to today's date, to include changes that have been made to the objects since the runtime version was created.
3. Generate the runtime version.

### Result

A build number is automatically assigned to a runtime version. If you regenerate a runtime version because some objects have been changed, the build number increases.

## Loading Data for a Runtime Version

## Loading Data for a Runtime Version

### Purpose

To create a runtime version for a knowledge base, you collect all the objects that are in a knowledge base at a certain point in time.

There are restrictions for the following objects:

- Actions are only loaded if you set the *Incl. actions* indicator when you create the runtime version. These actions are converted internally to procedures, so that you can define the sequence in which they are processed. Actions are inserted before any procedures, so actions are processed first in the SCE.
- Actions and procedures for characteristics and characteristic values are not loaded. If you want to use these dependencies, assign them to the configuration profile before you create the runtime version. You may need to add an IF condition to achieve the same effect.
- A runtime version only allows **one** configuration profile for a configurable material. If a configurable material has more than one configuration profile, you must select one.
- Value hierarchies are ignored. The value nodes in the hierarchy are filtered out, and only the individual values are loaded. For example, characteristic COUNTRY and value node 'Europe' are ignored, and only the individual values 'Italy', 'Denmark', and 'Portugal' are loaded.
- Characteristics with a user-defined data type are not loaded.
- All characteristics must be assigned to a class. This includes reference characteristics.
- If objects have been processed with engineering change management, the version that is valid on the date when you create the runtime version is loaded.
- You cannot use the built-in functions \$SUM\_PARTS and \$COUNT\_PARTS in dependencies. Instead, the SCE uses aggregating characteristics.
- Variant matching is not supported.

### SCE Mode

When you create a runtime version, you see the *SCE mode* indicator. This indicator is not active in Release 3.1I. In later releases, this indicator determines whether you load a knowledge base created with the R/3 configurator or a knowledge base created with the SCE.

If you use the SCE to create a knowledge base, there are no single dependencies and no configuration profiles. Instead, you define tasks and events for the knowledge base object. This changes some processes in the diagrams below.

In Release 3.1I, you cannot use this SCE functionality. Instead, you use compatibility mode, which lets you download existing configuration models to the SCE – with the restrictions described above. For this reason, this documentation does not contain any further detail of SCE mode.

### Process Flow



## Loading Data for a Runtime Version

Diagrams are used to show the process of loading the knowledge base. One box in the diagram denotes one action. An empty box shows that the action is repeated until all objects are loaded.

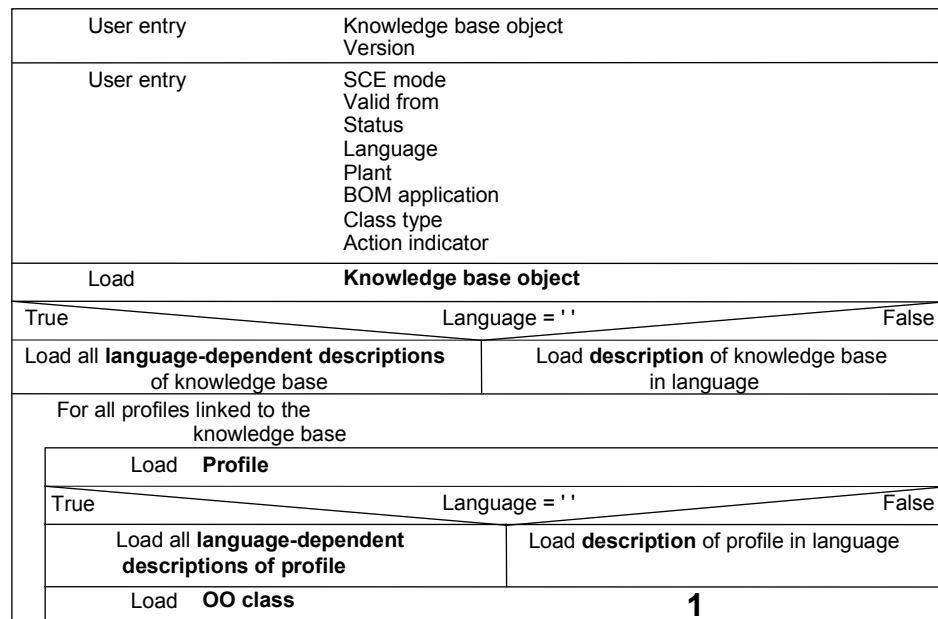
IF conditions (TRUE, FALSE) are shown in triangles in the boxes.

The large figures show where individual boxes are expanded or collapsed. For example, the first diagram has a **1** on the right-hand side of the box labeled 'Load OO class'. This box is expanded in the second diagram, which has a **1** at the top left-hand corner.

1. When you generate the runtime version, first the knowledge base object is loaded, with all its profiles.

If you enter a language for the runtime version, all the descriptions are loaded in this language. If you do not enter a language, the descriptions are loaded in all languages.

The OO class – either a configurable material or a class of configurable materials – is loaded for each profile.



2. The OO class can be an R/3 class or a material. The descriptions of the classes or materials are loaded. Depending on your entry in the runtime version, the descriptions are loaded in all languages or only one language.

The transaction for loading R/3 classes is described in point 4.

Materials are checked as to whether they are configurable or not.

If an OO class is a configurable material, the system checks for configuration profiles. If the material has more than one configuration profile, you must select one manually. The configuration profile you select is loaded.

The system checks the settings for BOM explosion. If the configuration profile has the setting *Planned/production order: Single-level*, the BOM is not loaded, because the BOM

## Loading Data for a Runtime Version

is not needed for configuration in the SCE. If the configuration profile has any other setting, the BOM is loaded.

Class allocations are loaded for both configurable and non-configurable materials.

Any values assigned to a material in these classes are loaded.

1 Load OO class			
True		Language = ''	
False			
Load all <b>language-dependent descriptions</b> of OO class		Load <b>description</b> of OO class in language	
True		Type of OO class = Material	
False			
True		Configurable material	
False			
False		SCE mode	
True			
Configuration profile exists?			
False			
True			
User: selects a configuration profile			
Load this <b>configuration profile 3</b>			
Scenario=Single-lev.			
True			
False			
		<b>6 Load BOM</b>	
True		Class type exists?	
False			
For each superior class linked to the material in the specified class type			
Load <b>SAP class 4</b>			
Load <b>assigned values</b> for material for characteristics of superior class			
		Load <b>SAP class 4</b>	

3. When you load the configuration profile, the system checks all of its dependencies.

If you set the *Incl. actions* indicator, the actions for the configuration profile are loaded and converted to procedures. If this indicator is not set, only the dependency nets and procedures for the configuration profile are loaded.

3 Load Configuration profile			
For all dependencies for the configuration profile			
True		Dependency <> Action or Action indicator = true	
False			
False		Constraint net	
True			
Load <b>classic dependency 5</b>		Load <b>dependency net 2</b>	

4. R/3 classes are loaded if a configurable material (KMAT) is allocated to classes, if the OO class is a class, or if the BOM contains class items.

All characteristics of the class are loaded.

The system then checks whether any of the classes are variant functions or variant tables, rather than actual classes, because both variant functions and variant tables are stored internally as classes in releases up to and including 4.0C. The characteristics

## Loading Data for a Runtime Version

assigned to variant tables and variant functions are loaded. Then the value assignment alternatives are loaded. Table entries in tables are also loaded.



Only the interface is loaded for variant functions, not the ABAP function code. For the SCE, you must create a JAVA method with the same name and exactly the same interface as the ABAP function module.

If the classes are neither functions nor variant tables, the selection conditions and preconditions for the characteristics are loaded.

All values of these characteristics are loaded, together with the preconditions allocated to the values.

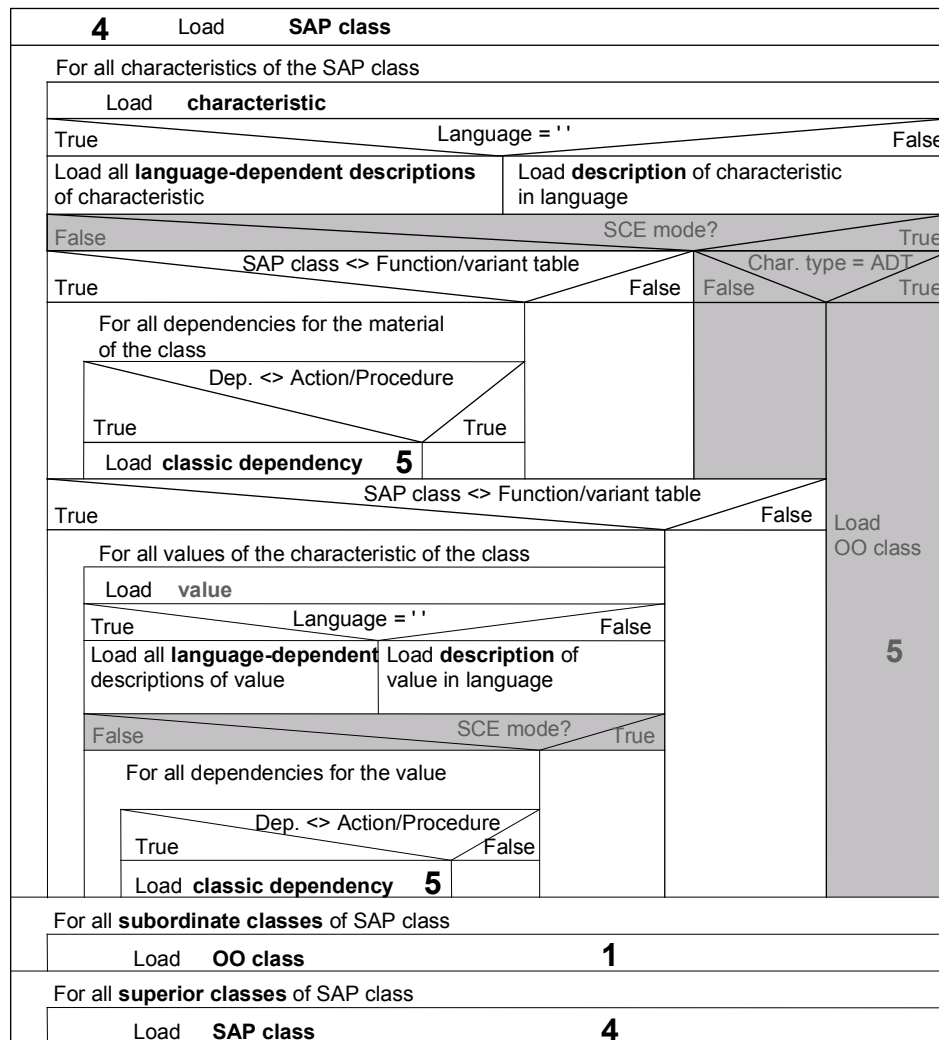


Actions and procedures for characteristics and characteristic values are not loaded.

You can allocate further classes or materials to the class that is loaded as the OO class.

This class can also have superior classes, from which it inherits characteristics. These superior classes are loaded, too.

## Loading Data for a Runtime Version



5. The BOM of configurable materials is loaded, unless the configuration profile has the setting *Planned/production order: Single-level*.

For materials with the configuration profile setting *Sales order*, only the sales-relevant items are loaded, because only these are relevant to the SCE. If the configuration profile has any other setting, the entire BOM is loaded.

BOM items can be either materials or classes. When you load the OO class, the system decides whether each BOM item is a class (class items in the BOM), a configurable material, or a non-configurable material. Class items are only loaded if they contain materials (class type 200). Class items for documents are not loaded (class type 201).

The selection conditions and preconditions for BOM items are loaded. If you set the *Incl. actions* indicator, the actions are loaded.



Loading Data for a Runtime Version

6 Load BOM (for plant and BOM application)		
True		Scenario <> 2 (Order Set)
False		
For all BOM items (classes or materials) for the material		For all sales-relevant items
Load OO class 1		Load OO class 1
False		SCE-Modus?
True		
For all loaded BOM items		
For all dependencies for BOM		
True		Dep. <> Action or Action indicator = true
False		
Load classic dependency 5		

6. For a dependency net, language-dependent descriptions are loaded for the net and all its constraints, in all languages or in the language you select.

The functions and tables used in the syntax are also loaded.

6 Load Dependency net		
True		Language = ''
False		
Load all language-dependent descriptions of net		Load descriptions of net in language
Load all language-dependent documentation of net		Load documentation of net in language
For all dependencies for material		
True		Language = ''
False		
Load all language-dependent descriptions of dependency		Load description deof dependency in language
Load all language-dependent documentation of dependency		Load documentation of dependency in language
For all functions for dependency		
Load function (SAP class) 4		
For all variant tables for dependency		
Load table (SAP class) 4		

7. For a single dependency, language-dependent descriptions and documentation are loaded in all languages or in the language you select.

The functions and tables used in the syntax are also loaded.

## Loading Data for a Runtime Version

<b>5</b> Load <b>Classic dependency</b>	
True	Language = '' False
Load all <b>language-dependent descriptions</b> of dependency	Load <b>description</b> of dependency in language
Load all <b>language-dependent documentation</b> of dependency	Load <b>documentation</b> of dependency in language
Load <b>compilation</b> for dependency	
For all functions for dependency	
Load <b>function</b> (SAP class)	<b>4</b>
For all variant tables for dependency	
Load <b>table</b> (SAPclass)	<b>4</b>

## Creating a Database Schema for the SCE

### Prerequisites

Before you can download objects in a knowledge base to your PC or laptop, you must create an empty database with your database tool once. To do this, you create an ODBC source, which you continue to use.

### Procedure

1. Create an empty database.
2. Assign an ODBC source name that points to the empty database.
3. Choose *Knowledge base* → *SCE database schema* → *Create on PC* to fill the database with the SCE tables.
4. Enter the name of the ODBC source, the user, and the password.

### Result

The database is filled with the database schema of the SCE. You can then download runtime versions to your PC or laptop.

---

**OO Class**

## OO Class

In the Sales Configuration Engine (SCE), an OO class can be a material or a class from the R/3 System.

Materials that have a material master in the R/3 System are, like classes, carriers of certain attributes that are inherited by instances of the material.



A material master record describes material Bolt\_Z. The actual bolts that are ordered with the material number are instances of material Bolt\_Z.

In this sense, the material is a class that defines the attributes of its instances.