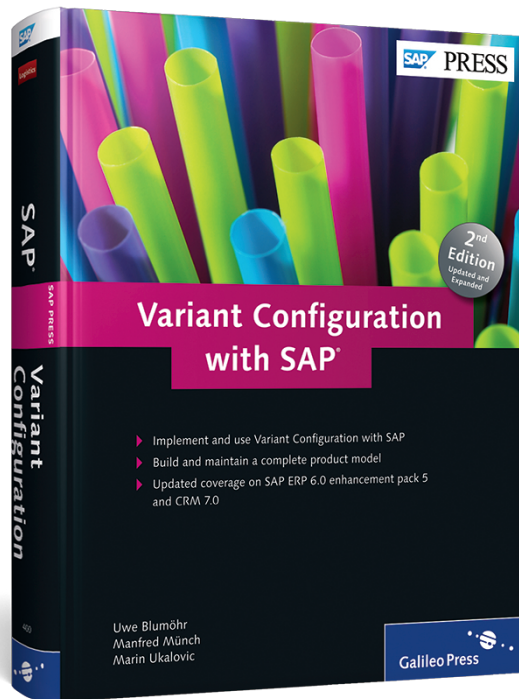


Uwe Blumöhr, Manfred Münch, and Marin Ukalovic

Variant Configuration with SAP®



 Galileo Press®

Bonn • Boston

Contents at a Glance

1	Basic Principles of Variant Configuration	31
2	Creating a Product Model for SAP Variant Configuration	69
3	Business Processes in SAP ERP	219
4	Customizing SAP ERP for Variant Configuration	311
5	Special Features of Product Configuration in SAP CRM	337
6	Challenges in Variant Configuration	367
7	Enhancements in SAP Industry Solution DIMP	447
8	Enhancements and Add-Ons in the SAP Partner Environment	465
9	Project Lead Reports on Projects and Project Structures	543
10	Customer Reports on the Introduction of SAP Variant Configuration	579
11	Configuration Workgroup	637
12	Outlook for SAP Business ByDesign	647
A	Database Tables of Variant Configuration	663
B	APIs of Variant Configuration	669
C	User Exits of Variant Configuration	671
D	Comprehensive Examples of Variant Functions	673
E	The Authors	681

Contents

Foreword	13
Introduction	21

1 Basic Principles of Variant Configuration 31

1.1	What Is Product Configuration?	31
1.1.1	Terminology	32
1.1.2	Elementary Configuration Modules	36
1.1.3	Product Configuration in Logistic Scenarios	39
1.1.4	Core Problem of Variant Diversity	41
1.1.5	Procedural and Declarative Approaches	44
1.2	What Is SAP Variant Configuration?	47
1.2.1	Product Configuration Using Variant Configuration (LO-VC)	48
1.2.2	Further Areas of Use	48
1.2.3	"Hello World" Example	49
1.2.4	Variant Configuration (LO-VC)	55
1.2.5	Internet Pricing and Configurator (IPC)	59
1.3	Enhancing Business Processes with Variant Configuration	63
1.3.1	Prerequisite for the Usage of Variant Configuration	63
1.3.2	Factors for the Usage of Variant Configuration	64
1.3.3	Exemplary Consideration on the Master Data Volume ...	66
1.4	Summary	67

2 Creating a Product Model for SAP Variant Configuration 69

2.1	Overview of the Modeling and Integration of Variant Configuration	69
2.1.1	Multivariant Product without Variant Configuration	70
2.1.2	Multivariant Product with Variant Configuration	70
2.2	Tools from the Classification System	75
2.2.1	Characteristic Management	75
2.2.2	Class Management	82
2.2.3	Classification	84
2.2.4	Search	85

2.3	Material Master, BOM, and Routing	87
2.3.1	Material Master of the Configurable Material	87
2.3.2	Super BOM of the Configurable Material	91
2.3.3	Super Task List for the Configurable Material	94
2.4	Configuration Profile and Configuration Scenarios	97
2.4.1	Overview of the Configuration Profile	97
2.4.2	Configuration Profile in Detail	99
2.4.3	Overview of Configuration Scenarios	105
2.4.4	Planned/Production Order without BOM Explosion Scenario	105
2.4.5	Order BOM Scenario	107
2.4.6	Sales Order (SET) Scenario	113
2.4.7	Planned/Production Order with BOM Explosion Scenario	118
2.5	Overview of Object Dependencies	121
2.5.1	Types of Object Dependencies and Assignment	121
2.5.2	The Procedural and Declarative Character of Object Dependencies	126
2.5.3	Global and Local Object Dependencies	126
2.5.4	Status of Object Dependencies	127
2.5.5	Object Dependencies in Classification and Variant Configuration	128
2.5.6	Execution Sequence of Object Dependencies	128
2.5.7	Basic Syntax Rules	131
2.5.8	Syntax Elements	134
2.5.9	Variant Tables and Functions	136
2.5.10	Evaluation Function for Object Dependencies	138
2.6	Object Dependencies for the Value Assignment Interface or the Sales View	142
2.6.1	Product Modeling Environment PMEVC	142
2.6.2	Example	146
2.6.3	Variant Tables in Detail	152
2.6.4	Constraints in Detail	158
2.6.5	Preconditions	164
2.6.6	Selection Conditions	167
2.6.7	Procedures	168
2.6.8	Reference Characteristics	171
2.6.9	Variant Functions	174
2.6.10	User Interface Design	177

2.7	Object Dependencies for BOM and Routing	179
2.7.1	Local and Global Object Dependencies	179
2.7.2	Selection Conditions for BOM and Routing	182
2.7.3	Class Nodes in BOMs	183
2.7.4	Classified Materials in BOMs	187
2.7.5	Procedures in BOM and Routing	189
2.8	Pricing for Configurable Materials	192
2.9	Product Costing for Configurable Materials	198
2.10	Material Variants	200
2.10.1	Material Master of the Material Variant	202
2.10.2	BOM and Material Variant	204
2.10.3	Routing and Material Variant	205
2.10.4	Pricing and Material Variant	207
2.10.5	Material Variant Matching	207
2.10.6	Material Variant Matching at the Header and Assembly Levels	211
2.11	How to Create a Product Model for the IPC	212
2.12	Summary	218

3 Business Processes in SAP ERP 219

3.1	Introduction—Variant Configuration in Business Processes	219
3.1.1	BOMs in Variant Configuration	219
3.1.2	Order Engineering Workbench	223
3.2	Variant Configuration with iPPE—Modeling	229
3.2.1	Product Variant Structure and Product Designer	230
3.2.2	Modeling From Requirements to Production	231
3.2.3	Feature and Requirement Structures	233
3.2.4	Structure Nodes, Component Variants, and Object Dependencies	235
3.2.5	Concepts	238
3.2.6	Filter: Explosion and Configuration Simulation	240
3.2.7	BOM Converter	242
3.2.8	PLM WebUI	245
3.3	Integration of Variant Configuration—The Classic Process	252
3.3.1	Sales Activities	253
3.3.2	Requirements Planning	255
3.3.3	Controlling the Requirements Transfer	256

3.3.4	Procurement: In-House Production or External Procurement	259
3.4	Processes with Extended Integration Aspects	260
3.4.1	In-House Production Process	262
3.4.2	Quality Management and Variant Configuration	265
3.4.3	Purchasing and Configurable Model Service Specifications	272
3.4.4	Project System, Configurable Standard Networks, and Variant Configuration	275
3.4.5	Customer Service and Configurable General Maintenance Task Lists	279
3.5	Planning and Variant Configuration	283
3.5.1	Excursus: Evaluations in the Variant Configuration Environment	284
3.5.2	Planning and Variant Configuration	288
3.5.3	Pure Assembly Planning	289
3.5.4	Characteristics Planning and Standard Product Planning	290
3.5.5	Characteristics Planning and Standard Product Planning with Long-Term Planning	296
3.5.6	Variant Planning and Planning with Planning Variants ...	299
3.5.7	Variant Configuration and SCM APO	304
3.5.8	Planning and SCM APO	307
3.6	Summary	308

4 Customizing SAP ERP for Variant Configuration 311

4.1	Explicit Customizing of Variant Configuration	311
4.1.1	Maintenance Authorizations	312
4.1.2	Status	313
4.1.3	Groups	315
4.1.4	Configurable Objects	316
4.1.5	Configuration User Interface	316
4.2	Classification System Customizing	317
4.3	Business Process Customizing Relevant for Variant Configuration	324
4.3.1	Configurable Material Master	324
4.3.2	Item Categories and Their Determination	327

4.3.3	Requirements Types, Requirements Classes, and Their Determination	329
4.3.4	Planning Strategies	332
4.3.5	Change Profiles in Order Change Management (OCM)	334
4.4	Summary	336

5 Special Features of Product Configuration in SAP CRM 337

5.1	Product Configuration in Different Channels	337
5.2	Configuration of Products versus Services	338
5.3	Procedure for Integrated Production in SAP ERP	339
5.3.1	Sales Configuration versus Production Configuration	340
5.3.2	Replication of the Master Data from SAP ERP	340
5.4	Creating a Product Model Using the PME	343
5.4.1	Essential Properties and Differences Compared to Modeling in SAP ERP	343
5.4.2	Calling the PME	344
5.4.3	Product Models versus Knowledge Bases	345
5.4.4	Version and Status Management	345
5.4.5	Classes, Characteristics, and Values	346
5.4.6	Object Dependencies in the PME	348
5.4.7	Transport of Knowledge Bases	355
5.5	IPC User Interface	355
5.5.1	JavaServer Pages and J2EE Engine	355
5.5.2	Extended Configuration Management (XCM)	356
5.6	Special Functions of the IPC User Interface	357
5.6.1	Images and Other Objects	357
5.6.2	Import-Export of Configuration Results	357
5.6.3	Pricing Overview	358
5.6.4	Better Handling of Restrictable Characteristics	360
5.6.5	Search/Set	360
5.6.6	Displaying Long Texts (as of SAP CRM 2006s)	361
5.6.7	Messages Controlled by the Configurator (as of SAP CRM 2006s)	361
5.6.8	Configuration Comparison (as of SAP CRM 2006s)	362
5.7	UI Designer (as of SAP CRM 7.0)	363
5.8	Summary	365

6	Challenges in Variant Configuration	367
6.1	Performance Optimization	368
6.1.1	Performance Bottlenecks—Occurrence and Influencing Factors	368
6.1.2	Reasons for Performance Bottlenecks	370
6.1.3	Performance Analysis	373
6.2	Change Management	376
6.2.1	Engineering Change Management (ECM)	376
6.2.2	Order Change Management (OCM)	388
6.3	Complex System Configurations	394
6.3.1	System Configuration—Definition	394
6.3.2	Dynamic Modification of the BOM Structure	395
6.3.3	Interlinked Configuration Structures in LO-VC	400
6.3.4	Composition Problems in SCE Advanced Mode	403
6.4	Master Data Distribution with Product Data Replication (PDR) ...	410
6.4.1	Challenge and Opportunities	411
6.4.2	PDR Components (ALE, Configuration Management, and Workflow)	413
6.4.3	Setting Up PDR	413
6.4.4	Preparations in the System	415
6.4.5	Setup and Customizing of PDR	418
6.4.6	Replication of a VC Model with PDR	427
6.5	Summary	444
7	Enhancements in SAP Industry Solution DIMP	447
7.1	Overview	447
7.2	DIMP—Discrete Industries and Mill Products	448
7.3	Special Requirements of the Mill Industry	449
7.3.1	Sales Order Processing and Production Scenarios	450
7.3.2	Production Discrepancies—Planned Configuration and Actual Configuration	450
7.4	Product Configuration Enhancements in SAP for Mill Products ...	452
7.4.1	Fast Entry of Characteristics—Simplified Entry of Configurable Document Items	453
7.4.2	Inheritance in Item Documents—Global and Local Items	456

7.4.3	Copying Default Values from the Customer Material Information Record	457
7.4.4	Working with Sales Order Versions	457
7.4.5	Variant Configuration in Connection with Make-to-Stock Production	460
7.4.6	Order Combination with Configurable Products	462
7.5	Summary	463

8 Enhancements and Add-Ons in the SAP Partner Environment ... 465

8.1	Sybit Model Tester (Company: Sybit GmbH)	467
8.1.1	Manual Testing—Transaction CU50	467
8.1.2	Benefits of Automated Tests	467
8.1.3	Sybit Model Tester	468
8.1.4	Summary	473
8.2	Sybit Configuration Visualizer (Company: Sybit GmbH)	473
8.2.1	Problem	473
8.2.2	Sybit Configuration Visualizer	475
8.2.3	User View	475
8.2.4	Modeler View—The Visualization Modeling Environment	477
8.2.5	System View	479
8.2.6	Summary	480
8.3	VCPowerPack (Company: AICOMP Group)	480
8.3.1	How VCPowerPack Works	481
8.3.2	VCPowerPack—CoreVC	481
8.3.3	VCPowerPack—SmartVC	482
8.3.4	VCPowerPack—SmartPR	483
8.3.5	VCPowerPack—SmartMD	485
8.3.6	VCPowerPack—Industry Solutions	485
8.3.7	Project Acceleration	485
8.3.8	Summary	486
8.4	it.cadpilot (Companies: itelligence AG and ACATEC Software GmbH)	486
8.4.1	CAD and SAP—Two Configuration Worlds?	487
8.4.2	Structure of Modern 3D CAD Systems	487
8.4.3	Controlling CAD Systems	488
8.4.4	Super BOM in Variant Configuration	489

8.4.5	Architecture	489
8.4.6	CAD Configuration	491
8.4.7	Advantages of a CAD Configuration Integrated into SAP ERP	492
8.4.8	Application Scenarios	493
8.4.9	Additional Options	495
8.5	Convenience Features for Sales, Marketing, and Modeling (Company: encoway GmbH)	497
8.5.1	K-Select	498
8.5.2	K-Assistant	500
8.5.3	K-Connect	501
8.5.4	K-Document	503
8.5.5	Quoteassistant	506
8.5.6	Summary of Convenience Features	507
8.6	top flow Framework and top flow-Variant Engine (Company: top flow GmbH)	507
8.6.1	Optimizing the Configuration Dialog Box	508
8.6.2	Functional Enhancements	511
8.6.3	New Object-Dependency Logic Options	512
8.6.4	Process Optimization with the top flow Variant Engine	514
8.7	Product Model Validation with ConfigScan (Companies: Fysbee SA and eSpline LLC)	515
8.7.1	Business Scenarios That Motivate the Need for Change ...	516
8.7.2	Anti-Patterns in Common Use	517
8.7.3	How ConfigScan Addresses these Issues	518
8.7.4	ConfigScan Validation Suite—The Basics	521
8.7.5	Working with the Test Editor	521
8.7.6	Use Case: Nokia Siemens Networks	524
8.7.7	Summary	526
8.8	Managing Variant Configuration (Company: eSpline LLC)	526
8.8.1	Managing the LO-VC Model Lifecycle	528
8.8.2	Managing the LO-VC Transactional Processes	534
8.8.3	Summary	539
8.9	Summary	540

9 Project Lead Reports on Projects and Project Structures 543

9.1	"We're Implementing SAP!"—A Project Lead's Experience Report	543
9.1.1	The Marketing Pitch and What Will Follow—Clarify the Prerequisites for Your Work	544
9.1.2	Analyze Your Business Processes and Improve Them	546
9.1.3	How Many Instances Would You Like to Have?	547
9.1.4	The Regional versus Global Approach	549
9.1.5	Dealing with Modifications to the Standard System	550
9.1.6	The Compromises You Can or Cannot Accept	551
9.1.7	Finding the Appropriate External Support	553
9.1.8	Communicate Changes Effectively	555
9.1.9	Communicate Necessary Compromise Effectively	556
9.1.10	Train Your Employees	557
9.1.11	Problems After Going Live	558
9.1.12	Changing Mass Data	559
9.1.13	Changing Business Models	561
9.2	Roles in a Variant Configuration Team	562
9.2.1	Expertise and Experts	562
9.2.2	Putting Together and Structuring the Project Team	567
9.3	ASAP for Variant Configuration Projects	567
9.3.1	Project Preparation	569
9.3.2	Business Blueprint	570
9.3.3	Realization	570
9.3.4	Final Preparation	571
9.3.5	Go-Live and Support	572
9.3.6	Golden Client Approach	572
9.3.7	Specific Features of IPC Scenarios	574
9.4	Summary	576

10 Customer Reports on the Introduction of SAP Variant Configuration 579

10.1	Progress of the Project at Getriebebau NORD	580
10.1.1	Initial Situation	581
10.1.2	Measures	582
10.1.3	Results	586

10.1.4	Summary	588
10.2	Configurable Materials at Krones AG	590
10.2.1	Project	590
10.2.2	Results	590
10.2.3	Summary	593
10.3	Progress of the Project at Hauni Maschinenbau AG	594
10.3.1	Personnel Resources	595
10.3.2	Result	596
10.3.3	Using the Order Engineering Workbench	599
10.4	Variant Configuration at the Felix Schoeller Group	602
10.4.1	Project	602
10.4.2	Results	604
10.4.3	Extending Variant Configuration Using the IPC	608
10.4.4	Summary	610
10.5	SAP at Hülsta and in the Hüls Corporate Group	610
10.5.1	Initial Situation	611
10.5.2	Preparation	611
10.5.3	Project Objectives and Results	612
10.5.4	Summary	619
10.6	Lenze Group—Past, Present, and Future Configuration	620
10.6.1	Present Configuration—The EuLe Project	620
10.6.2	Future Configuration—Powerful Process Integration	624
10.7	Product Configuration at Baldor Electric	626
10.7.1	Starting Point of the Project	626
10.7.2	Key Characteristics of the Project	627
10.7.3	Basics of the Variant Model	633
10.7.4	Conclusion	634
10.8	Summary	635

11 Configuration Workgroup 637

11.1	Introduction to the CWG	637
11.2	Tasks and Objectives	638
11.3	History	640
11.4	Organizational Structure	642
11.5	CWG Conferences	643
11.6	CWG Portal	644
11.7	CWG Sandbox System	645
11.8	Summary	646

12 Outlook for SAP Business ByDesign	647
12.1 SAP Business ByDesign	647
12.2 Product Configuration in Medium-Sized Businesses	648
12.3 Make to Order in SAP Business ByDesign	650
12.3.1 Extending the Product Concept	651
12.3.2 Make to Specification	652
12.3.3 Lightweight Product Variants	653
12.4 Product Configuration in SAP Business ByDesign	654
12.4.1 Product Model	654
12.4.2 Product Properties	655
12.4.3 Integration of a Configurator	657
12.4.4 Process Automation	658
12.5 Summary	658
Appendices	661
A Database Tables of Variant Configuration	663
B APIs of Variant Configuration	669
C User Exits of Variant Configuration	671
D Comprehensive Examples of Variant Functions	673
E The Authors	681
Index	683

- ▶ A list of all BOM explosions including assigned object dependencies.
- ▶ An evaluation of the class nodes.
- ▶ A detailed list of the characteristic value assignment including assigned object dependencies for characteristics and values.
- ▶ An evaluation of the configuration profiles, including object dependencies.

2.6 Object Dependencies for the Value Assignment Interface or the Sales View

As already described, object dependencies are required for two usages: the high-level configuration (sales configuration, in the dialog, for the value assignment interface) and the low-level configuration (BOM and routing explosion, also without dialog). The following section discusses the first usage in more detail.

2.6.1 Product Modeling Environment PMEVC

Various transactions or methods are provided for the maintenance of object dependencies for the value assignment interface. The most important maintenance environment for object dependencies for the value assignment interface, both local and global, is the PMEVC product modeling environment (see Figure 2.29). This section therefore focuses on this environment. Section 2.7 introduces additional maintenance options.

The PMEVC product modeling environment has been available as Transaction PMEVC since SAP ERP Release 5.0. A similar function is also part of the IPC. PMEVC is short for *Product Modeling Environment Variant Configuration*. The concept behind this transaction is to create an environment in which you can maintain the entire variant model via the model structure from the high-level configuration perspective. Similar to the CUMODEL variant model browser, you can first obtain an overview of the existing model structure and then navigate to details.

You can also create and change numerous components of the configuration model from this product modeling environment. This enables you to create and change all types of object dependencies, both global and local. The same applies to configuration profiles, variant tables, and IPC data.

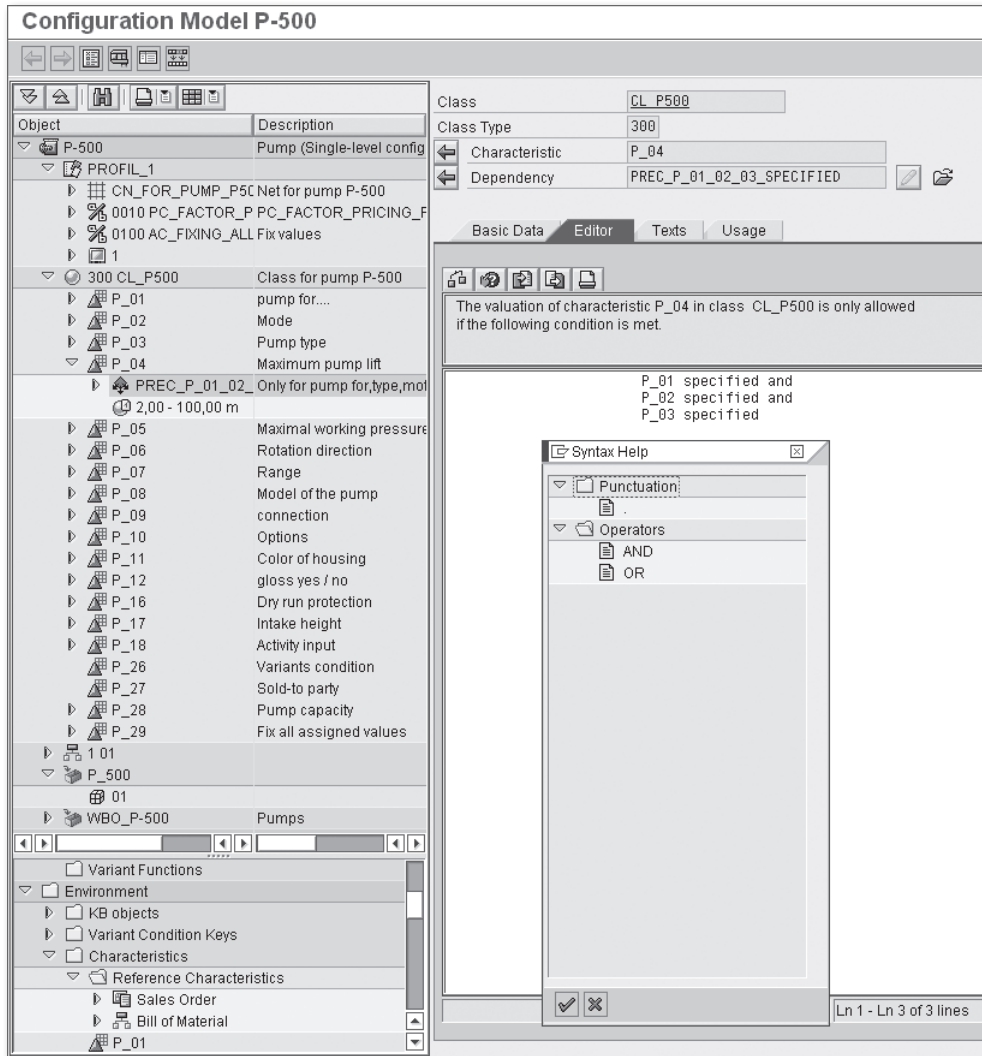


Figure 2.29 Product Modeling Environment PMEVC

The product modeling environment uses an additional editor for the maintenance of object dependencies. In contrast to the traditional object dependency editor, this editor allows you to use the following elements:

- ▶ Context-sensitive input help
- ▶ Drag-and-drop

- ▶ Object dependency wizard for preconditions, selection conditions, and table-based constraints

You can call the context-sensitive input help via the function key `F4` or the second button in the editor (see Figure 2.29). All types of syntax elements as well as characteristics and characteristic values are provided. You can insert them in their respective positions in the object dependency syntax.

The drag-and-drop function enables you to copy individual characteristic values from the lists on the left (see Figure 2.29) to the editor in PMEVC. As the example in Listing 2.7 shows, the characteristic and the characteristic value are transferred in the form of an equation.

```
char1 = 'value1'
```

Listing 2.7 Example of a Syntax Generated via Drag-and-Drop

As already mentioned, the object dependency wizard enables you to create preconditions and selection conditions as well as table-based constraints without having to write the syntax yourself. The wizard queries all necessary information, and the system creates the syntax and all other required data.

The PMEVC product modeling environment not only enables you to completely maintain all object dependencies that are relevant for the value assignment interface; PMEVC also allows for nearly all maintenance steps of the modeling for the high-level configuration. This includes the following aspects:

- ▶ Maintenance (creation, modification, display, assignment) of object dependencies for the configuration profile, characteristics, and characteristic values
- ▶ Class-specific characteristic adaptation
- ▶ Simple classification (no multiple classification)
- ▶ Creation of a configuration profile (several profiles for one material master are not possible, but you can maintain all existing configuration profiles)
- ▶ Usage of change numbers
- ▶ Maintenance of the structure of variant tables
- ▶ Maintenance of the content of variant tables
- ▶ Modification of object dependencies for the BOM
- ▶ Creation of the knowledge base and runtime version for the IPC

- ▶ Material-specific activation of the IPC as the configurator in SAP ERP
- ▶ Maintenance of the user interface design
- ▶ Maintenance and assignment of variant conditions for pricing

Enhancement Package 5 (EHP5) of ERP Release 6.0 is likely to provide the additional option to not only change existing object dependencies in the BOM but also create new object dependencies. You can also use drag-and-drop.

The system functions are also supposed to be enhanced. If you start the simulation from PMEVC (☒ button in Figure 2.29), the system first displays a screen for assigning values to reference characteristics. Depending on the configuration scenario—that is, whether it is a sales order or material variant scenario—you can assign values to all relevant reference characteristics before the actual value assignment screen opens.

Besides these two aspects (creation of new object dependencies for BOM items and simulation with reference characteristics), EHP5 will probably provide the following additional PMEVC functions:

- ▶ Creation of new characteristic values
- ▶ Creation of new descriptions and long texts in additional languages for characteristics and characteristic values
- ▶ Easier maintenance of characteristics groups within the scope of user interface design, including drag-and-drop option
- ▶ Detailed view for BOMs and BOM items

Most of the master data of the Variant Configuration model cannot be maintained via PMEVC. Consequently, it is required that the following master data be created via the common transactions in advance:

- ▶ Characteristics
- ▶ Classes
- ▶ Material masters
- ▶ BOMs including all objects at the item level
- ▶ Routings including all objects at the operation level
- ▶ Variant functions

- ▶ Change numbers
- ▶ Objects that couldn't be created in PMEVC before Release ERP 6.0

After this introduction of PMEVC, the following section describes an example of its usage.

2.6.2 Example

After these rather theoretical explanations, it may be helpful to provide you with a practical example of how you can use PMEVC to create object dependencies for the value assignment interface. For this purpose, you use the object dependency wizard in PMEVC.

To map dependencies between the individual characteristics of the value assignment interface, you should use tables or, if they don't become too long, variant tables. The advantage of tables is that you can read the dependency type from them more easily than when evaluating the syntax of the object dependencies directly. Furthermore, the usage of variant tables has a major advantage if the model is "alive": If the dependencies in the model change, you have to change only the content of the variant table, without having to modify the syntax of the object dependencies.

The easiest way to evaluate the table is to write object dependencies that query the table and only allow for value assignments that comply with the table. This is to be implemented in such a way that no disallowed value assignments are possible. The list of the allowed values for each characteristic is supposed to be dynamically restricted in such a way that only allowed value assignments are possible. For the selection of such "elegant" object dependencies, you must use a type for which the user doesn't have to specify a point in time when the object dependencies are to be processed. You should therefore use constraints.

In Figure 2.30, PMEVC was called with material T-VPC. The system has found the BOM for the material. These are initially the two only entries in structure (❶). You now require at least the variant class and the configuration profile. As already mentioned, the variant class, including its characteristics, needs to be created outside PMEVC. In Figure 2.30 the existing variant class is already assigned.

This variant class (or a complete group of variant classes) was previously included in the PMEVC environment (see the bottom left of Figure 2.31, here under ENVIRONMENT • CLASSES AND CONTEXT menu). You then simply assign the variant class via drag-and-drop.

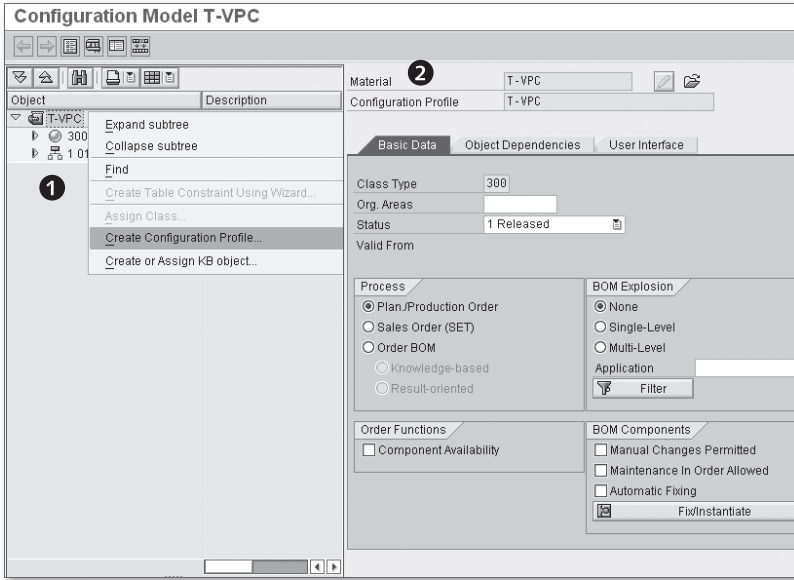


Figure 2.30 Getting Started in PMEVC: Class Assignment and Creation of a Configuration Profile

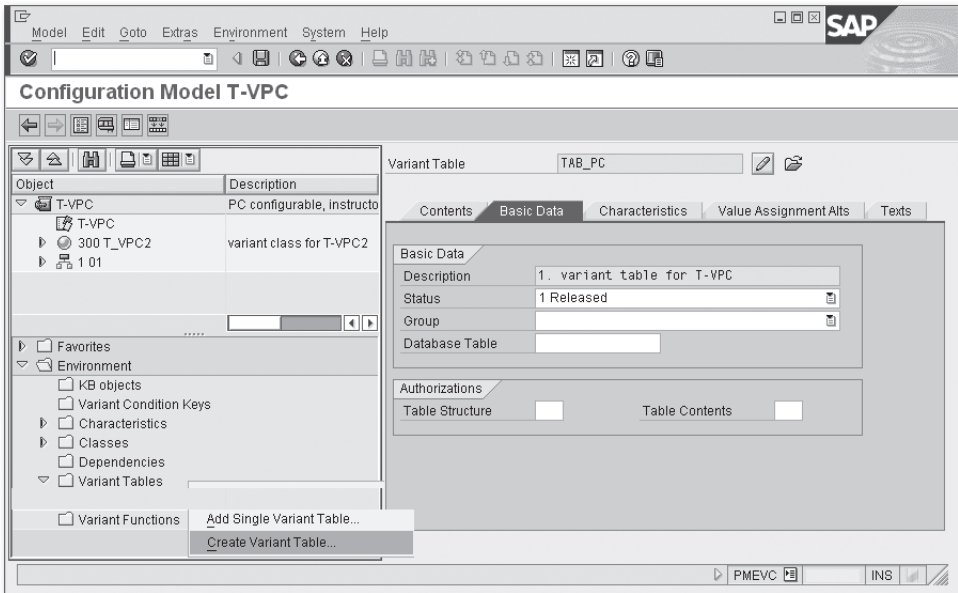


Figure 2.31 Variant Tables—Creation and Content Maintenance with PMEVC

In contrast to the variant class, you can create the configuration profile directly from PMEVC. As you can see in Figure 2.30 (2), this function is available in the context menu at the material master level. Similar to the common transaction for the creation of configuration profiles, you can create a configuration profile with the default values. Unlike the common transaction, there are also default values for the name and variant class type.

In addition to the two previously mentioned steps, the model structure consists of four objects: material master, configuration profile, variant class, and BOM.

A variant table (see Figure 2.31) is supposed to map the allowed combinations for the value assignment of numerous characteristics. In this example, we assume that this includes the three characteristics: "Special wish," "Casing," and "CPU." For this purpose, perform the following steps:

1. **Create a variant table**

First you create the variant table via the context menu in the environment, because it isn't provided here yet.

2. **Name of the table**

The window CREATE VARIANT TABLE opens, in which you enter a name for the table. ENGINEERING CHANGE MANAGEMENT is optional and not supposed to be used in this example.

3. **Description of the table**

The system then displays the detail screen (see Figure 2.31) with five tabs. In the BASIC DATA tab, enter a description (language-dependent), and then release the variant table.

4. **Assigning characteristics**

In the CHARACTERISTICS tab, specify the three mentioned characteristics in any sequence.

5. **Entering the table content**

Finally, enter the allowed combinations of the value assignment with regard to the three characteristics as rows in the CONTENTS tab.

After the variant tables have been created and their content has been maintained, you require object dependencies. Object dependencies read the table and dynamically

restrict the value lists of the corresponding characteristics in such a way that only value assignments from the table are possible. For this purpose, start the table constraint wizard via the context menu for the configuration profile, as shown in Figure 2.32.

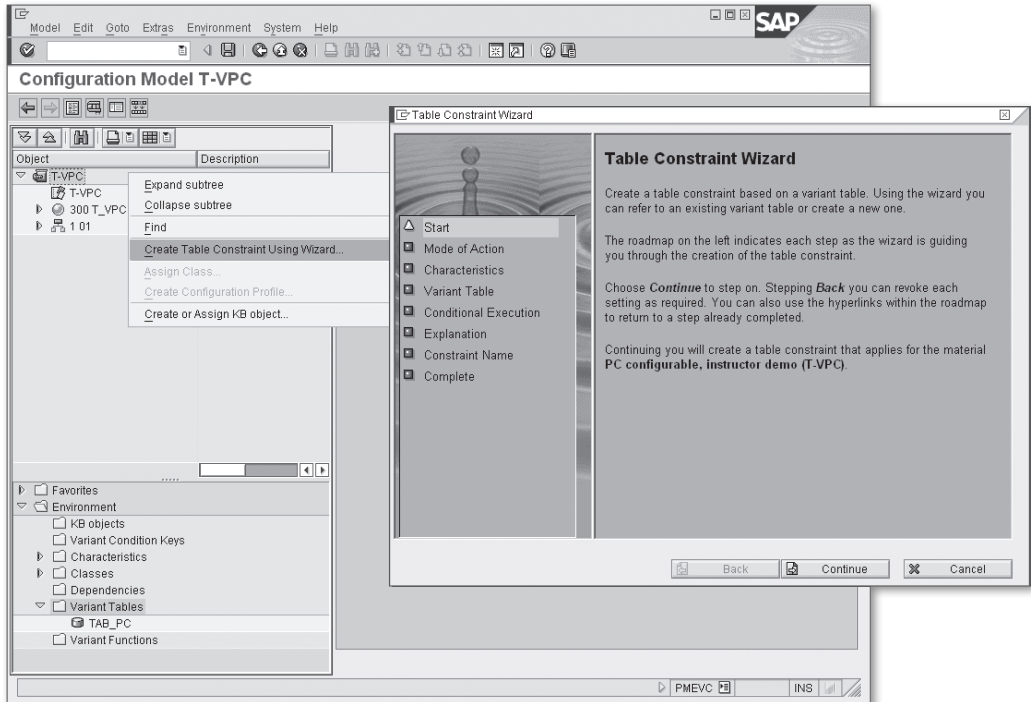


Figure 2.32 Creating Object Dependencies Using the Table Constraint Wizard

The wizard guides you through the individual steps of the creation of constraints and queries you for all required information. It is possible that the wizard may dynamically adapt the steps to the already specified information. The wizard processes the following steps:

1. Start

The first step of the wizard provides information on the procedure for the creation of a constraint with reference to a variant table.

2. **Mode of Action**

The second step queries you about the mode of action. Here, you're supposed to restrict the value lists for characteristics. This is the first selection option in this step. The difference from other selection options is described in the following.

It is possible that the wizard won't provide the VALUE RESTRICTION option. In this case, the required prerequisite—that the characteristics whose value lists are supposed to be restricted are indicated as restrictable in the characteristic definition—is not met.

3. **Configuration Object**

Next, the wizard queries you for the configuration object. In addition to characteristics from classes, you can also use and evaluate other reference objects in constraints. Because this example is supposed to restrict characteristics of a variant class, the corresponding variant class is also the configuration or reference object.

4. **Variant table**

After the configuration object step, the variant table selection is implemented. Because this example contains only one variant table in the PMEVC environment, no comprehensive selection can be made.

5. **Explanation**

In the EXPLANATION step, you can assign a language-dependent long text to the object dependencies, that is, to the table constraint. It also lists the characteristics of the variant table to exclude characteristics for object dependencies if required.

6. **Constraint Name**

For constraints, the name must be assigned externally, that is, by the user. This request, including a short text, is made in the CONSTRAINT NAME step.

7. **Complete**

All steps are completed. A description of these steps is provided once again in detail before the user can complete the table constraint.

The system then displays the completed table constraint (see Figure 2.33). After saving, you can directly test it by calling the configuration simulation from PMEVC via the TEST button.

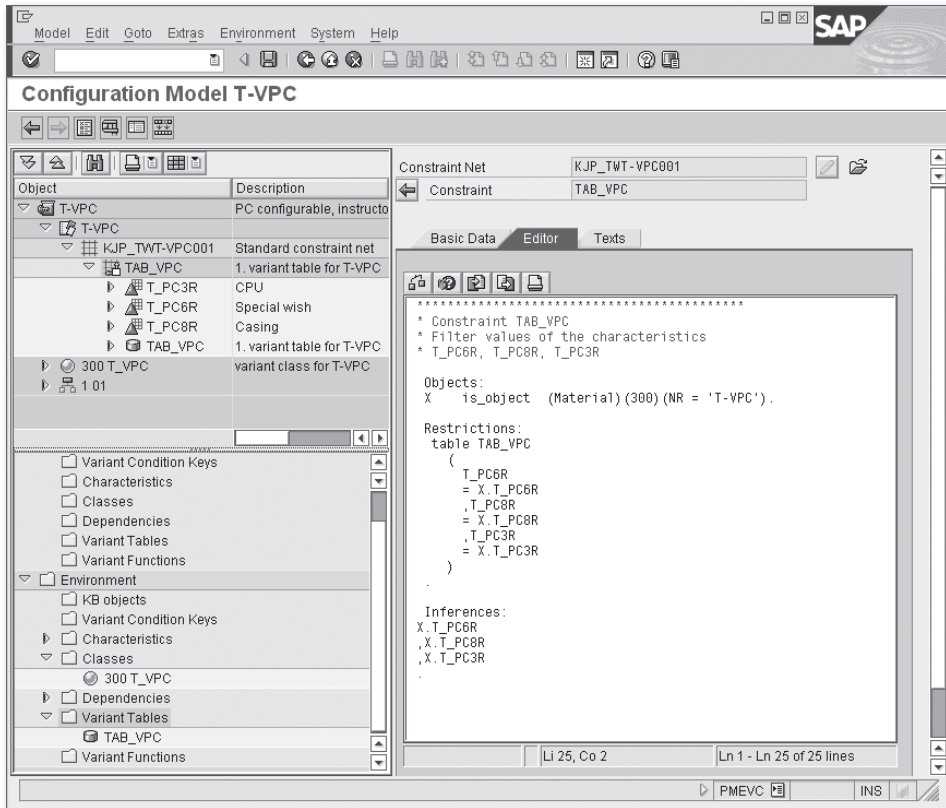


Figure 2.33 Result of the Table Constraint Wizard

The value assignment interface displays the three characteristics: “Special wish,” “Casing,” and “CPU.” Initially, the corresponding input help (**F4** key) provides all values from the variant table for each characteristic. However, if you start assigning values to any of the three characteristics, the system provides the values only for the other two characteristics that lead to an allowed value assignment according to the variant table. The same applies to two characteristics to which values have been assigned.

A special feature of the traditional configurator is that if the allowed value range for a characteristic is restricted to a value, the system automatically sets this value. This is done only for required characteristics in the IPC. Note that the characteristics need to be restrictable according to the characteristic definition. As a result, the

lists of the allowed values for characteristics to which values have been assigned are restricted to the value assignment, as already described.

2.6.3 Variant Tables in Detail

The first usage of variant tables was introduced in the example above. You can address variant tables in all types of object dependencies. The columns of variant tables are always characteristics. The rows represent value assignment combinations. You can use variant tables for different purposes:

- ▶ Value restrictions in constraints (in this context, values can already have been assigned—as described in the example)
- ▶ Inferences of values in constraints or procedures
- ▶ Conditions as preconditions, selection conditions, if conditions in procedures or constraints, and as a condition part in constraints
- ▶ Consistency checks via constraints

These purposes are discussed in more detail in the introduction to the corresponding types of object dependencies in Sections 2.6.4 through 2.6.7.

At this point, the structure and maintenance of variant tables will be introduced (see Figure 2.31 and Figure 2.34). You can maintain the variant table's structure and content via PMEVC. There are also specific transactions for the following tasks:

- ▶ Content maintenance (Transaction CU60)
- ▶ Creating, changing, and displaying the table structure (Transactions CU61 through 63)

The BASIC DATA tab of variant tables (see Figure 2.31) contains names, the description (language-dependent short text), the status, and the group. The status can be adapted via Customizing and also includes the content maintenance and usage in object dependencies as well as distribution locks for the content and structure.

The same aspects apply to groups as to characteristic and class groups; it is also a separate list in Customizing. Furthermore, you can couple the variant table with a database table in the basic data. This is described in detail later in this section. The basic data is complemented by the authorization groups for the content and structure maintenance.

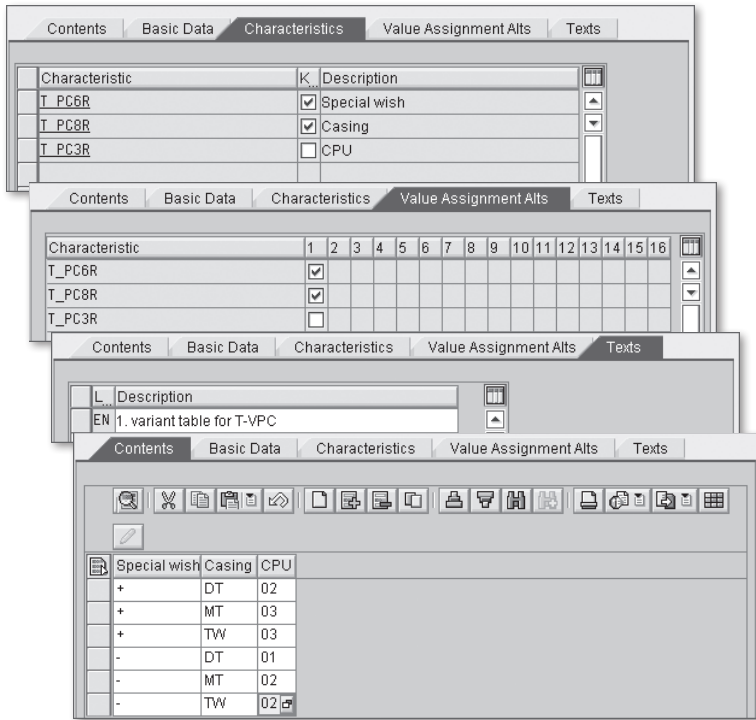


Figure 2.34 Structure and Maintenance of the Variant Table (PMEVC)

The CHARACTERISTICS tab, which is assigned to a variant table, provides the columns of the variant table. For the maintenance of the table content and for the usage in object dependencies, the settings in the characteristic definition—such as single-level/multilevel, restrictable, required characteristic, default values, or object dependencies—are irrelevant. In the characteristic view, you can define a first key. This key is merely a prerequisite and is significant only if you want to infer values from the variant table. This can be done via constraints or procedures. For value restrictions, conditions, or mere consistency checks, the system ignores the key information.

For inference of values, constraints with an inference part can evaluate more than the VALUE ASSIGNMENT ALTERNATIVE (key in the CHARACTERISTICS tab) that is specified in the tab. You can create these additional alternatives in the corresponding view.

In PMEVC, you can also maintain the elements of the CONTENT directly in the variant table.

Besides maintaining content from PMEVC, you can implement the content via a common transaction, namely, Transaction CU60. In addition to the standard display (❶ in Figure 2.35), this transaction also allows for displays as a matrix (❷) and as a list (❸). These last two displays enable you to easily decide which combination is supposed to be used (decision table).

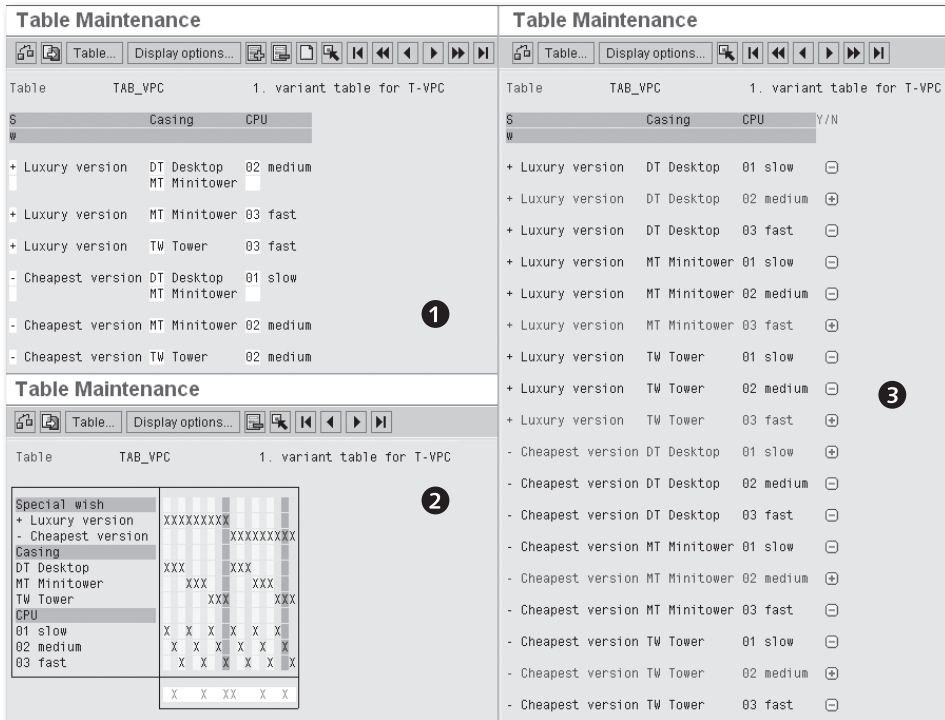


Figure 2.35 Variant Tables—Content Maintenance with the Common Transaction

!! IPC-Compatible Table Content

You can also select multiple values for each field in the standard display. This display isn't IPC-compatible, but you can transfer it to such a display using the respective menu entry (Add up, Untag).

Besides the already introduced options of maintaining variant table content using PMEVC or a common transaction, namely, Transaction CL60, you can also import content from Excel tables. For this purpose, you can use Transaction CU60E.

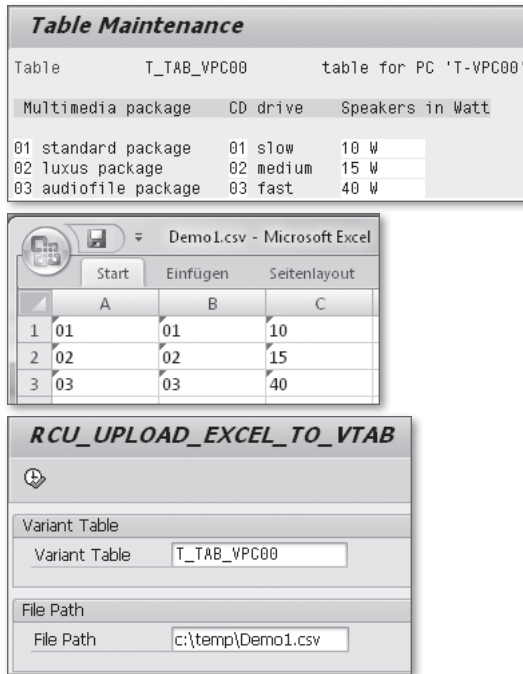


Figure 2.36 Transaction CU60E—Data Import from Excel Tables into Variant Table

To import data using Transaction CU60E, the SAP ERP system requires an existing variant table. This means that you first have to create the required characteristics and then use these characteristics to create the variant table structure.

The example in Figure 2.36 assumes that the result of the data import is a variant table (as illustrated at the top of Figure 2.36). The specified preparations in the ERP system include the creation of the three characteristics that are used in this example (multimedia package, CD drive, and speakers) as well as the creation of the variant table structure (T_TAB_VPC00).

When creating and filling the Excel table, the following needs to be considered: The columns of the Excel table and their sequence must correspond to those of the variant table, and all values in the Excel table must be in the *text* format. This also applies to numerical characteristics in the variant table. The Excel table mustn't contain headers or descriptions—only language-independent characteristic values. The system checks the format of the values during the data import and cancels the transaction if necessary. The Excel file needs to be saved as a *csv* file (see Figure 2.36). This means that semicolons are used as separators for this Excel file format.

Transaction CU60E requires solely the name of the variant table as well as the name and address of the Excel file as specifications. You start the data import using the EXECUTE button (F8). Only complete data imports are possible; an option for change uploads does not exist. The system deletes possibly existing variant table content and fills the variant table with the new entries.



Transaction CU60E

With regard to Transaction CU60E, please refer to SAP Note 516885. It also contains a documentation for this transaction.

As already mentioned, you can use variant tables in all types of object dependencies. Basically, the syntax always looks the same (see Listing 2.8).

```
table <table name>
(<column characteristic>=<interface characteristic>,...)
```

Listing 2.8 Syntax Concept for Calling Variant Tables

You can break down this visual similarity as follows:

- ▶ The call starts with the keyword `table`.
- ▶ Then the name of the table is specified.
- ▶ Next, all columns of the variant table that are to be evaluated are listed in brackets. You don't always have to evaluate the entire table. Because the characteristics of the table don't have to correspond to the characteristics of the user interface, the system assigns the corresponding characteristic to the value assignment interface after each column characteristic.

Figure 2.33 shows an example of such a table call in constraints. Note that `∗.` precedes the characteristics from the value assignment interface. This refers to material T-VPC, as you can see in the constraint directly above the table call. As mentioned in Chapter 1, you cannot address objects (the material master in this case) as flexibly as in constraints. In this case, you can only use the `$self.`, `$root.`, or `$parent.` levels. The table access from Figure 2.33 would then be as shown in Listing 2.9.

```
table TAB_VPC ( T_PC6R = $self.T_PC6R,
               T_PC6R = $self.T_PC6R,
               T_PC6R = $self.T_PC6R)
```

Listing 2.9 Table Access in Preconditions, Selection Conditions, and Procedures

Characteristics of the value assignment interface for which the values are supposed to be inferred must be linked with `$self.`; `$parent.` and `$root.` are also allowed.



Figure 2.37 Coupling of Variant and Database Tables

Coupling with database tables also needs to be discussed (see Figure 2.37) in the context of the maintenance of the variant tables. The background for this function is the requirement to also address database tables in object dependencies. However, this is not directly possible in object dependencies. There are two options for addressing database tables in object dependencies.

► **Directly addressing a database table (variant function)**

You create a variant function and address the database table in the function module that corresponds to the variant function. The variant function is used in object dependencies.

► **Indirectly addressing a database table (coupled variant table)**

You create an appropriate variant table and couple it with the database table. If the coupling is active, you can directly include the variant table in any type of object dependencies. Then the database table is also addressed.

The second option uses an assignment of the database table in the maintenance of the basic data of the variant table structure. You can also distinguish between the following two scenarios.

1. Scenario 2—Starting Point Database Table

An existing database table is to be addressed using this approach. You must create the appropriate characteristics, that is, characteristics with the appropriate format, in this scenario. You have to create characteristics only for the fields (that is, the columns of the database table) that are to be addressed in the object dependencies. Then you create a variant table that contains exactly these characteristics as columns. You must include the name of the database table in the basic data of the variant table. This allows for a field assignment. All characteristics of the variant table are linked to the columns of the database table. Finally, you must activate the coupling. Afterward, you can evaluate the database table in all types of object dependencies by addressing the coupled variant table in the syntax.

2. Scenario 2—Starting Point Variant Table

An existing variant table that may already be used in object dependencies is to be converted into a database table. This scenario is used, for example, when variant tables reach a size that may lead to performance problems. However, there may also be other reasons.

In this case, you need to create an appropriate database table. With regard to the format, the fields of the table must be created in such a way that they can include at least the values of the corresponding characteristics. The key fields of the database table are not relevant for Variant Configuration. Only the key combination of the variant table is critical for the evaluation of object dependencies. Of course, you must select the database table key in such a way that uniqueness is ensured for later content.

Analogous to the first scenario, you must now couple the variant table with the still empty database table, activate it, and assign the corresponding fields. Finally, the content of the variant table is transferred to the database table using Transaction CU59. In this second scenario, you don't have to adapt all object dependencies that use this variant table. They have exactly the same function as before. Bear in mind that the content of the variant table is inactive when the coupling has been activated.

In the standard version, you maintain the content directly in the database table. You can lock the content maintenance of the variant table via the status. You can also delete the content of the variant table to prevent misunderstandings.

Additional maintenance in the variant table with a delta transfer of this data to the database table is also possible. However, this isn't very useful because you cannot delete rows when implementing a delta transfer, and problems may occur due to the key of the database table.

2.6.4 Constraints in Detail

As already mentioned, you can use all types of object dependencies for the configuration in the value assignment interface. Constraints are the most important and default type of object dependencies, because they can be used for nearly all tasks. You can therefore perform the following tasks using constraints:

- ▶ Setting values
- ▶ Checking values (in consistency checks)

- ▶ Addressing any objects that are also in the multilevel configuration (not only \$self, \$parent, or \$root)
- ▶ Working with a high performance level
- ▶ Working in a declarative way (this means you don't have to consider a processing sequence or similar factor in the modeling process)

Constraints are collected in dependency nets (also referred to as constraint nets). Dependency nets are generally assigned to the configuration profile. It doesn't matter whether a large number of constraints are assigned to a dependency net. However, the number of dependency nets in a configuration model should be kept to a minimum, even though you cannot always ensure that only one dependency net is assigned to the configuration profile of the header material of the configuration profile.

Need for Multiple Constraint Nets

[Ex]

If, for example, individual configurable assemblies of the overall model are sold separately, all constraints that are related to this assembly must also be assigned to the configuration profile of this assembly via a dependency net.

Within the dependency net, the constraints are local object dependencies; the dependency net itself is global. Note that both have a status. Constraints are consequently active only if the following conditions are met:

- ▶ The constraint is released.
- ▶ The dependency net is released.
- ▶ The dependency net is assigned to a configuration profile that is considered during configuration.

A constraint consists of at least two and at the most four sections. Objects and restriction are the mandatory sections here. The following explains the four sections according to their sequence:

1. Objects

In this section, you enter all used classes and objects and define variables.

2. Condition

This section is only used to (optionally) specify a central condition under which the constraint is supposed to be evaluated.

3. Restriction

In this section, you define equations, tables, and functions for inferences of values and/or value checks.

4. Inferences

This section enables you to (optionally) enhance the inference and restrict characteristic values.

The objects section is mandatory and must contain the objects that are addressed in the constraint. Objects can be classes, material masters, and documents. The declaration has the following syntax:

► **Class**

(<class type>)class name, for example, (300)T_VPC

► **Material master**

(Material)(<class type>)(Nr = '<material number>'),
for example, (Material)(300)(Nr = 'T-VPC')

► **Document**

(Document)(<class type>)(Type = '<document type>', Version = '<document version>', Part = '<document part>', Nr = '<document number>'),
for example, (Document)(017)(Type='DRM',Version='01',Part='000',Nr='T-VPC')

In the objects section, you can also locally define variables for the constraint; the constraint in Figure 2.33 uses such an object variable. Object variables for classes are declared by `is_a`, and for all other objects by `is_object`, as shown in Figure 2.33. The constraint in Figure 2.33 would be as shown in Listing 2.10 without variables.

```
objects:      (300)t_vpc
restrictions: table tab_vpc ( t_pc6r = (300)t_vpc.t_pc6r,
                             t_pc8r = (300)t_vpc.t_pc8r,
                             t_pc3r = (300)t_vpc.t_pc3r).
```

Listing 2.10 Sample Constraint: Table Call without Using Variables

Note that in this context you can also declare the variant class instead of the material master in the objects sections, as done here. After the equals sign, the syntax is as follows: <object>.<characteristic>.

In the objects section, you can also define characteristic variables, which are class-specific. You therefore don't have to use object variables here. In the syntax, this

is implemented with `where` and a list of the variables separated by a semicolon. The constraint in Figure 2.33 would be as follows with characteristic variables (Listing 2.11):

```
objects:      (300)t_vpc where  ?spe = t_pc6r ;
                                ?cas = t_pc8r ;
                                ?cpu = t_pc3r .
restrictions: table tab_vpc (  t_pc6r = ?spe,
                                t_pc8r = ?cas,
                                t_pc3r = ?cpu).
```

Listing 2.11 Sample Constraint—Table Call Using Characteristic Variables

Specific Variable Names

As shown here, question marks are often used as variables so that they can be easily identified. You cannot use variables for characteristics from tables, that is, for columns.

[+]

If the objects section lists multiple objects, use a comma to separate them.

The restriction section is the second section that is mandatory in constraints. Here, you can carry out consistency checks. The constraint reports an inconsistency when the restriction section is not met. You can also use constraints to infer values. For this purpose, the equations must be solved for the value that is supposed to be inferred, or you must use the inference section. This is further detailed in the context of the inferences section below. The restriction section enables you to restrict values (see Figure 2.33). You can use variant tables and variant functions.

You can also work with conditions in the restriction section. The `if` syntax element is provided for this purpose. Additional syntax elements, such as `then` or `else`, are not available. As shown in Listing 2.12, first the statement and then the `if` condition is provided.

```
restrictions:
(300)t_vpc.m1 = 'a'      if (300)t_vpc.m2 = 'x1',
      false             if (300)t_vpc.m2 = (300)t_vpc.m3.
```

Listing 2.12 Restrictions with `if` Conditions

This restriction section contains two statements that are listed with a comma. The first statement consists of the assignment of the `a` value for the `m1` characteristic. The second statement only leads to an inconsistency message if the `if` condition

after `false` is met. The processing of the `false` syntax element basically generates inconsistency messages and can only be used in constraints.

In constraints, you can use a condition section. The condition section must always be inserted between the objects and the restriction section. It contains exactly one logical expression. The constraint is not processed until the condition of the condition section is met, which is very good regarding the performance. You can also work with variables and variant tables in the condition section, as Listings 2.13 and 2.14 show.

```
objects:   (300)t_vpc where   ?os = t_pc01;
                                   ?hd = t_pc04,
           (300)t_vpr where   ?tr = t_pr02.
condition : specified ?hd.
restrictions : ?tr = ?os
```

Listing 2.13 Sample Constraint with Condition Section

```
objects:   (300)t_vpc where   ?spe = t_pc6r;
                                   ?cas = t_pc8r;
                                   ?cpu = t_pc3r.
condition:   table tab_vpc (   t_pc6r = ?spe,
                                   t_pc8r = ?cas,
                                   t_pc3r = ?cpu).
restrictions: false.
```

Listing 2.14 Sample Constraint with Table Call in the Condition Section

In Listing 2.13, the restriction section is evaluated under the condition that a value has been assigned to the `t_pc04` characteristic. In this case, the system checks whether the same values have been assigned to the `t_pc01` and `t_pr02` characteristics. If no values have been assigned to `t_pc01`, the system copies the value assignment of `t_pr02` to this characteristic.

In Listing 2.14, an inconsistency message is output if the `(300)t_vpc` variant class in the value assignment interface corresponds to a row of the `tab_vpc` variant table in the configuration. This is used if inconsistent value assignment combinations are collected in variant tables. You can also create such constraints with the table constraint wizard. In this case, you must select the `CHECKING INCONSISTENT COMBINATIONS` entry as the mode of action. In this context, have a look at Figure 2.32 in Section 2.6.2.

The inferences section, which is optional just like the condition section, is the fourth section of a constraint. This section is always the last section of the constraint and enhances the evaluation of the restriction section. This "enhanced evaluation" can refer to equations, variant tables, variant functions, and restrictable characteristics. The syntax of the inferences section is merely a list of characteristics.

Consequently, an equation, $V = L * W * H$, for example, in a restriction section without a subsequent inferences section is only evaluated for calculation in such a way that V is the product of L , W , and H . However, if the constraint has the following structure, the fourth value is inferred from any three values (see Listing 2.15).

```
objects:   (300)t_vpc where
           v = t_pc91 ;   l = t_pc92;
           b = t_pc93 ;   h = t_pc94.
restrictions: v = l * b * h.
inferences:   v, l, b, h.
```

Listing 2.15 Sample Constraint with Equation and Inferences Section

Variant tables and functions for which more than one value assignment alternative is defined are additional examples (see Figure 2.34 in Section 2.6.3). If you don't use the inferences section here, the system can only evaluate the first key, that is, the first value assignment alternative. However, if the constraint has the following structure and if two additional value assignment alternatives exist for the first and third characteristics and for the second and third characteristics, the system can infer the remaining third characteristic from the table from any two characteristics to which values have been assigned (see Listing 2.16).

```
objects:   (300)t_vpc where   ?spe = t_pc6r ;
                               ?cas = t_pc8r ;
                               ?cpu = t_pc3r.
restrictions: table tab_vpc ( t_pc6r = ?spe,
                               t_pc8r = ?cas,
                               t_pc3r = ?cpu).
inferences: ?spe, ?cas, ?cpu.
```

Listing 2.16 Sample Constraint with Variant Table and Inferences Section

For usage with restrictable characteristics, have a look at the constraint in Figure 2.33 in Section 2.6.2.

Now that we've discussed constraints in detail, the following sections deal with the other types of object dependencies. As already mentioned, you can use all

types of object dependencies to design the value assignment interface in sales and distribution.

2.6.5 Preconditions

You can use preconditions to disallow individual characteristic values or entire characteristics for the value assignment interface. If you don't use preconditions, values can be assigned to any characteristic of the value assignment interface in any sequence. You can select any value from the list of the allowed values—irrespective of the value assignment of other characteristics. In this context, you have to find answers to two questions:

1. What is supposed to be disallowed? That is, which characteristic or which characteristic value is supposed to be dynamically disallowed?
2. When is it supposed to be allowed? That is, when is the corresponding characteristic or characteristic value supposed to be allowed within the scope of the value assignment?

The storage location of the respective precondition answers the first question. The syntax answers the second question. For example, if the XYZ engine is only supposed to be offered for the sport version of a car configuration, a precondition must be assigned to the XYZ characteristic value of the characteristic for the engine selection (What is supposed to be disallowed?). The syntax contains the prerequisite that the sport version was selected for which the XYZ+ engine is allowed (When is it supposed to be allowed?).

What effect does it have when the `$self.version = 'sport'` precondition is assigned to the XYZ characteristic value?

► **Assigning the “Sport” value to the “Version” characteristic**

If the “Sport” value is assigned to the “Version” characteristic, the list of allowed values in the characteristic for the engine will provide all values as if no precondition exists.

► **Assigning a different value than “Sport” to the “Version” characteristic**

If a different value than “Sport” is assigned to the “Version” characteristic, the “XYZ” value will be missing in the list of allowed values in the characteristic for the engine.

► **No value assigned to the “Version” characteristic**

Note that the precondition is considered to be met if no value is assigned to the

"Version" characteristic. In this case, the list of allowed values of the engine characteristic would include all engine values. In the standard version all allowed values are initially available. During the value assignment, the system hides the values that are no longer allowed. If you don't want to use this standard logic, you must implement this by adding `$self.version = 'sport'` and `$self.version specified`. The system then first provides all values that can be generally selected, and the list of allowed values is gradually extended.

► **Sequence of the value assignment**

The precondition mentioned in the previous item is elegant in one direction only: if values are assigned first to the version and then to the engine. If you start by assigning values to the engine, you can select any engine and any version. Only then is the precondition evaluated. It retroactively disallows the XYZ engine. This leads to an inconsistency message if the XYZ engine and a version other than sport are selected. You can avoid this by assigning a precondition to the sport version. Another option is to force a processing sequence. A value is not supposed to be assigned to the "Engine" characteristic until the version is known. In this case, you use a precondition for the characteristics. You assign a precondition of the `$self.version specified` (When is it supposed to be allowed?) to the engine characteristic (What is supposed to be disallowed?).

► **Multiple preconditions**

You can also assign multiple preconditions to characteristic values or characteristics. In this case, the value assignment is only allowed if all preconditions are met. It can be considered an `And` link. You can only implement an `Or` link between preconditions when you include the conditions in a precondition. A precondition can be any complex condition using any brackets, negations, and concatenation with `and` and `or`.

Values or characteristics that are excluded via preconditions are not displayed in the value assignment interface by default. However, you can use the settings in the configuration profile (see Figure 2.38), for example, to define that disallowed characteristic values or characteristics are displayed but not used for the value assignment.

In Figure 2.38, the settings were called via the menu during the configuration. The `WITH EXCLUDED CHARACTERISTICS` checkbox was selected here. As a result, the disallowed value, XYZ, is displayed but cannot be selected. Similarly, disallowed characteristics would be displayed, but you couldn't assign values to them.

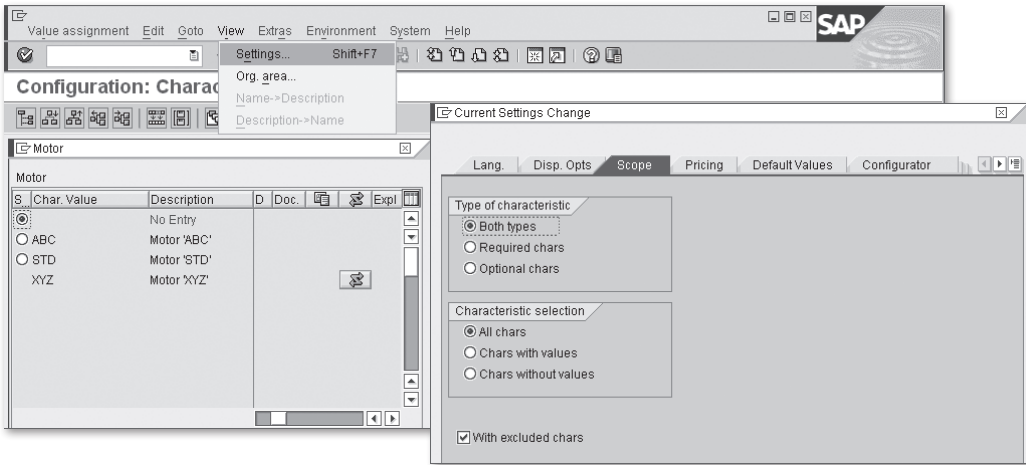


Figure 2.38 Displaying Excluded Characteristic Values (Preconditions)

You can use variant tables in all types of object dependencies, that is, also in preconditions. The syntax of a precondition that uses the table from Figures 2.31 and 2.34 could be, as shown in Listing 2.17:

```
table TAB_VPC (
    T_PC6R = $self.T_PC6R,
    T_PC8R = $self.T_PC8R,
    T_PC3R = $self.T_PC3R)
```

Listing 2.17 Sample Variant Table in a Precondition

This example has the same syntax as a procedure. Instead of `$self`, you can also use `$parent` or `$root` everywhere—provided that this is correct with regard to content. You could assign such a precondition with exactly this syntax to one of the three mentioned characteristics; however, this wouldn't be a clean solution.

Considering the increasing elegance of the solution, the following three options can be used to disallow values.

- ▶ Precondition for the characteristic (as described)
- ▶ Precondition for the characteristic value (as described later)
- ▶ A constraint with this variant table and restrictable characteristics (as the most elegant solution; see Figure 2.33)

If you selected the variant with the precondition for the characteristic, the precondition wouldn't affect the value assignment of the three characteristics. Not

until values have been assigned to all three characteristics addressed in the variant table does the precondition become active and output an inconsistency if the table doesn't contain this value assignment combination.

However, preconditions for characteristic values are more elegant than this variant with a precondition for the characteristic but still not as elegant as the variant with a constraint. For example, if you assume that values are assigned to the characteristics in a fixed sequence and the T_PC3R characteristic is the last characteristic to which a value is assigned, you can provide preconditions for the characteristic values of this characteristic. The precondition for the value '03' would have the following syntax (see Listing 2.18):

```
table TAB_VPC (
    T_PC6R = $self.T_PC6R,
    T_PC8R = $self.T_PC8R,
    T_PC3R = '03')
```

Listing 2.18 Sample Variant Table in a Precondition for Characteristic Value '03'

Analogous preconditions also apply to the other values of this characteristic. Compared to the first variant—that is, the precondition for the characteristic—this has the advantage that only the values that lead to a consistent value assignment are provided for the third characteristic. Preconditions for characteristics or characteristic values don't require restrictable characteristics.

2.6.6 Selection Conditions

Selection conditions can dynamically convert optional characteristics into required characteristics.

Optional Characteristics

This technology requires that the ENTRY REQUIRED checkbox in the characteristic is not set, that is, the characteristic is actually a so-called optional characteristic.



If you assign a selection condition to such a characteristic, the characteristic dynamically becomes a required characteristic if the condition is met. Let's illustrate this with an example.

[Ex]**Sport Version Requires a GPS**

For the car configuration, the selection of a GPS needs to become a prerequisite for the sport version. In this context, the rules and the syntax must be analogous to the preconditions. The selection conditions must be assigned to the GPS characteristic (What is required?); the content of the syntax is the condition of the sport version (When is it required?). The syntax is analogous to the precondition specified previously: `$self.version = 'sport'`. The selection condition with this syntax is assigned to the GPS characteristic. This way, the selection of a GPS is required for the sport version.

Compared to the rules of preconditions, there are two differences, which must be considered:

- ▶ **Multiple selection conditions for a characteristic**

If multiple selection conditions are assigned to a characteristic, it is sufficient for the characteristic to become a required characteristic when one selection condition is met.

- ▶ **No value assigned to a characteristic in a selection condition**

If a selection condition addresses characteristics to which no value has been assigned, the selection condition is not met. In contrast to a precondition with the same syntax, the selection condition above outputs "false," that is, not met if no version is selected.

2.6.7 Procedures

Procedures are object dependencies that set values. In contrast to other types of object dependencies, they can affect processing sequences. For this purpose, the configuration profile is assigned with procedures, and a sorting is transferred. There is also the option of assigning procedures to characteristics and characteristic values. However, this method is not supported for the configurator of the IPC and is consequently not discussed here. Instead, we'll discuss in detail the assignment to the configuration profile. Procedures allow for *fixed* or *dynamic* assignments of values. The following list describes in detail what this means:

- ▶ **Fixed assignment of values**

A fixed assignment of values refers to an assignment of values that cannot be overwritten by users or constraints. Attempts to do so would result in inconsistencies. Procedures can also not overwrite *external* assignments of values, that is, values set by users or constraints. The same applies to users and constraints.

Index

? → Assignment of default values, 136

|| → Character string link, 135

A

A2A → Application to application, 658

ABAP and object dependencies, 131

ABAP function module, 57, 591

ABAP programming language, 55, 564

Abstracter data type (ADT), 407

Access node, 230

Action, 61, 124

Actual characteristics, 451

Actual configuration, 450

Actual value assignment, 451

ACWG → American Configuration

Workgroup, 640

Adaptable Custom Solution (ACS), 224, 229

Advanced mode, 61

Advanced Planning & Optimization (APO),
304, 451

Aggregation, 408

Aggregation characteristic, 410

ALE → Application Link Enabling, 314

Alternative Dates, 383

American Configuration Workgroup (ACWG),
640

Americas' SAP Users' Group (ASUG), 637

Analysis tool, 141

AP Application Platform, 62

AP Configuration Engine, 62, 337, 338, 370

Application group, 455

Application Link Enabling (ALE)

ALE distribution, 411

ALE distribution model, 413, 418

ALE partner profile, 417, 418

ALE partner profiles, 413

Application log, 439

Application Programming Interface (API), 669

Application-to-application (A2A), 658

AP Pricing Engine, 338

Arithmetic operator, 134

ASAP Implementation Roadmap, 567

Assemble-to-order (ATO), 41

Assembly processing, 275, 393

Assignment of default values, 136

Assignment of production resources and tools,
97

Assignment of values

Dynamic, 123, 169

Fixed, 123, 168

ASUG → Americas' SAP Users' Group, 637

Asynchronous, 618

ATO → Assemble-to-order, 41

Authorization group, 77

Authorization object

C_LOVC_DEP, 312

C_TCLS_BER, 323

Automated product configuration, 35

Avenue from VC, 532

Avenue Migration, 533

Avenue Orchestrator, 527

Avenue Remodel, 533

Avenue to VC, 533

Avenue XML, 532

B

B2B → Business-to-Business, 48

B2C → Business-to-Consumer, 48

Backward chaining, 47

BAId, 229

Balancing, 408

Baldor Electric, 626

Baseline, 428, 430

Explosion, 432

Batch classification, 451

Batch job, 431

Batch selection criteria, 451

Bill of material explosion, 101

Bill of materials, 37, 214, 219, 371, 377

Application, 101, 222, 420

Configurable bill of materials, 38
Enhancements, 38
Explosion, 101
Filter, 101
Has part, 406
Maintenance, 183
Manual enhancement, 38
Maximum BOM, 38
Minimum BOM, 406
Part of, 400, 406
Relationship
Structures, 582
Super BOM, 38
Synchronization, 232
Usage, 221
 Business Object, 648
 Building block, 406
 Business Application Programming Interface (BAPI), 534
 BO → Business object, 648
 Business-to-Business (B2B), 48
 Business-to-Consumer (B2C), 48
 Business Transaction Event (BTE), 417

C

Cable, 449
 CCE → Core Constraint Engine, 657
 Change
 ECR, 378
 Mass data, 559
 Master, 378
 Master record, 378
 Number, 376, 378, 390
 Number type, 378
 Type, 380
 Changeable material variants, 450
 Change management for master data, 73
 Characteristic, 36, 37, 71, 371, 377
 Characteristic value, 371
 Create, 51
 Dependency, 37
 Disallow a characteristic value, 164
 Display, 455
 Fast data entry, 454
 Group, 77, 177

Management, 75
 Name, 76
 Planning, 292, 330
 Status, 77
 Value, 36
 Characteristics-dependent planning, 451, 461
 Character string link, 135
 Character string operator, 135
 Checkbox
 Configurable Material, 89
 Class hierarchy, 55
 Classification, 84
 System, 55, 75
 Classified materials, 187
 Class management, 82
 Class network, 55
 Class node, 57, 82, 183, 217, 322, 371, 591
 Cloth, 449
 Coaching project, 582
 Coil, 449
 Company philosophy, 552
 Comparison function, 226
 Component condition, 348
 Component formula, 348, 354
 Component list in planned and production order, 260
 Component structure, 37
 Component decomposition, 37
 Composition problem, 39, 395, 403
 Decomposition problem, 403
 Dynamic modification of the BOM structure, 395
 Interlinked component structure, 395, 400
 Top-down approach, 403
 Condition, 159, 162, 348
 Condition technique, 59
 Condition type, 193
 VA00, 193
 VA01, 193
 Configit A/S, 535
 ConfigScan Validation Suite, 515, 528
 Configurable
 Assembly, 102
 General maintenance task list, 280
 Material, 49
 Model service specification, 274
 Purchased material, 591

- Standard network*, 276, 278
 - Configuration, 35
 - Browser*, 103
 - Configuration result*, 35
 - Definition*, 427, 429
 - Folder*, 427, 429, 430
 - High-level configuration*, 58, 368, 378
 - Indicator*, 55
 - Interactive configuration*, 35, 45
 - Low-level configuration*, 58, 368, 378
 - Management*, 413, 419
 - Module*, 36
 - Profile*, 49
 - Result*, 35, 373
 - Scenario*, 99, 105
 - Settings*, 103, 209
 - Step*, 45
 - Task*, 32
 - User interface*, 316
 - Configuration model, 35
 - Complete*, 53
 - Configuration process
 - Order BOM*, 397, 398
 - Sales order*, 398
 - Configuration profile, 56, 72, 215, 377, 398
 - And PMEVC*, 146
 - Of the configurable component as a filter*, 259
 - Configuration rules, 37, 55
 - Declarative*, 400
 - Declarative approach*, 44, 46
 - Procedural approach*, 44, 45
 - Simple rule*, 45
 - Configuration-specific modules, 55
 - Configuration structure, 103
 - Bottom-up approach*, 403
 - Configuration supporting point, 294
 - Configuration UI, 316
 - Configuration Workgroup (CWG), 59, 637
 - Board of Directors*, 642
 - Bylaws*, 638
 - CWG conference*, 643
 - CWG Portal*, 638, 644
 - CWG Sandbox*, 645
 - Executive Committee*, 642
 - Membership*, 642
 - President*, 643
 - Registered association*, 642
 - Configurator, 36
 - Configurer, 594
 - Configure-to-order (CTO), 41, 340
 - Confirmation, 260, 269
 - Constraint, 46, 56, 124, 146, 586
 - Net*, 56, 159, 408
 - Constraints and class nodes, 186
 - Construction type, 281
 - Consultant role
 - IPC expert*, 564
 - Master data expert*, 565
 - Modeler*, 564
 - Pricing expert*, 565
 - Project lead*, 566
 - Solution architect*, 563
 - VC expert*, 563
 - Context-sensitive input help in PMEVC, 144
 - Conversion, 243
 - Copy control, 255
 - Core Constraint Engine (CCE), 657
 - Correction package, 428, 438
 - Coupling of variant table and database table, 152, 157
 - CTO → Configure-to-order, 41
 - Customer Relationship Management (CRM), 48, 337, 338
 - Customer Services, 261, 279
 - Customer-specific production, 40
 - Customizing of material master, 324
- ## D
-
- Database table, 152, 157, 586
 - AUSP*, 322
 - Database table and variant table*, 137
 - ESLL*, 274
 - PLPO*, 191, 278
 - STPO*, 189
 - TB31*, 317
 - VBAK*, 171, 253
 - Data type, 77
 - Character format CHAR*, 77
 - Numerical format NUM*, 77
 - Data volume, 67
 - Decision table, 154

Declarative approach, 44
 Declarative modeling, 348
 Declarative object dependencies, 126
 Decoupling point, 450
 Default value
 Dynamic, 169
 Delta list, 62, 342
 Dependency editor, 181
 Dependency Maintenance Table (DMT), 245
 Dependency net, 159
 Dependency type, 121
 Action, 61, 372
 Constraint, 56, 400
 Monitoring rule, 408
 Precondition, 55, 121
 Procedure, 56, 122, 372
 Reevaluating rule, 408
 Selection condition, 56, 57, 122
 Discrete Industries and Mill Products (DIMP), 447
 Distribution lock, 313
 Distribution order, 428, 433
 Package, 428
 Distribution package, 412, 428, 433
 Distribution type, 427
 Distribution unit, 428, 433
 Document, 82, 377
 Drag-and-drop in PMEVC, 144
 DSAG → Deutschsprachige SAP-Anwendergruppe, 637
 Dynamic assignment of values, 123, 169
 Dynamic bill of materials, 93
 Dynamic database (DDB), 374
 Dynamic instantiation, 39
 Dynamic sequence, 95

E

ECM history requirement, 385
 ECM → Engineering Change Management, 73
 Effectivity parameter, 240
 Emcien, Inc., 535
 EmcienMatch, 535
 EmcienMix, 535
 Engineering Change Management (ECM), 73, 376, 423, 432, 439, 591

Engineer-to-order (ETO), 32, 41, 223, 591, 596
 Equipment, 264, 279
 eSpline LLC, 466, 527
 ETO → Engineer-to-order, 32
 Event type coupling, 426
 Experience report, 543, 566
 Experts, 562
 Explosion profile, 420, 429
 Extended Configuration Management (XCM), 608
 External procurement, 259

F

False, 162
 Films, 449
 Filter, 259
 Finish-to-order, 450
 Fixed assignment of values, 123, 168
 Formula, 348
 Forward chaining, 46
 FOX → Framework for Object Explosion, 420
 Framework for Object Explosion (FOX), 420, 428, 430
 Function, 138, 176
 Module, 176, 587
 Formula, 348
 Function modules and object dependencies, 137
 Fysbee SAS, 466, 528

G

General maintenance task list, 262
 German-Speaking SAP User Group, 637
 Global object dependencies, 127, 179
 Gold VC Client, 528, 628
 Group, 315
 GSS PSM to BOM, 248
 Guided Structure Synchronization (GSS), 245, 248

H

Header, 96
 Header material, 591, 607
 Hello World example, 49
 Hierarchy allowed, 322
 High-level configuration, 101, 340

I

IDoc → Intermediate Document, 411
 Individual customization, 36
 Industry solution, 447, 449
 Inferences, 160, 163
 InfoSet, 285
 Info structure S138, 284
 Inheritance, 55
 In-house production, 259
 Initiating object records for OCM, 331, 390
 In list queries, 135
 Inspection characteristic, 265
 Inspection lot, 264
 Inspection type, 266
 Instantiation
 Dynamic, 39, 397
 Manual, 397
 Material variant, 399
 Integrated Product and Process Engineering (iPPE), 230, 232
 Access, 240
 Access node, 230
 BOM converter, 232, 242
 Concept, 232, 238
 Concept group, 239
 Configuration simulation, 240
 Enhanced, 237, 238
 Feature and requirement structure, 231, 233
 Filter, 240
 Node, 230
 Structure node, 230
 Variance scheme, 231, 238
 Variants, 231
 View node, 231
 Integrated product engineering, 232
 Integration of a configurator, 657

Interaction Center, 337
 Interface design, 102
 Intermediate Document (IDoc), 411, 413, 435
 Internet Pricing and Configurator (IPC), 48, 59, 370, 608, 647
 IPC_CONFIGURATION_UI, 316
 IPC database, 213
 IPC Data Loader, 214
 IPC modeling
 delta, 218
 IPC Product Configurator, 63
 Invisible, 172
 IPC → Internet Pricing and Configurator, 608
 iPPE → Integrated Product and Process Engineering, 230
 Item category, 93, 257, 327, 453
 AGC, 257
 TAC, 257, 327
 TAM, 327
 Item category determination, 327
 Item category group, 257

J

Java programming language, 59, 565

K

KBIF → Knowledge Base Interchange Format, 640
 KB object → Knowledge-base object, 213
 KBO → Knowledge-base object, 61
 KMAT, 50
 Knowledge base, 61, 341, 373, 404
 Knowledge Base Interchange Format (KBIF), 532, 640
 Knowledge-base object (KB object), 61, 213

L

Length orientation, 449
 Lifecycle phase, 419
 Lifecycle profile, 419
 Local object dependencies, 126, 179, 180

Lock, 99
 Logical operator, 134
 Logistics information system, 284
 LO-VC-compatible mode, 60
 LO-VC Variant Configuration, 48, 369, 370
 Low-level configuration, 340

M

Maintenance authorizations, 312
 Maintenance plan, 280
 Maintenance task list, 48
 Make-to-order, 40, 603
 Make-to-order (MTO), 40, 603
 Production, 461
 Make-to-specification, 652
 Make-to-stock, 39, 41, 603
 Production, 460
 Managing Variant Configuration, 526
 Manufacturing step, 460
 Mass customization, 36
 Mass production, 36
 Master data change management, 376
 Master inspection characteristic, 265
 Material, 49, 377
 Material BOM, 73
 Changeable, 450
 Material master, 71, 262
 Customizing, 325
 View concept, 88
 Material variant, 73, 201, 653
 Matching, 201, 207
 Maximum BOM, 38
 mdata, 136
 Message interface, 658
 Message type (IDoc), 411
 Middleware (SAP CRM), 214
 Mill industry, 449
 Mill products, 447
 Minimum BOM, 406
 MMCOM, 198
 MMCOM-VKOND, 198
 Model concept, 57
 Model service specification, 49, 260, 262, 274
 MRP
 Group, 90, 258

Type, 90
 MTO → Make-to-order, 40, 603
 MTS → Make-to-stock, 39, 41, 603
 Multiple BOM, 92, 221
 Multiple classification, 320
 Multivariant product, 70

N

NetWeaver Business Client (NWBC), 245
 Network, 262, 591
 Not specifiable, 349
 Number range, 417

O

Object dependencies, 55, 56, 73, 125, 378
 Basic data, 181
 Declarative, 126
 Global, 126
 Local, 126
 Procedural, 126
 Release, 127
 Semi-declarative, 126
 Status, 127
 Wizard in PMEVC, 144, 146
 Object dependency type
 Constraint, 372
 Object management record, 383
 Objects, 159
 Search in classes, 85
 Type, 422
 OCM → Order Change Management, 73, 260, 334
 On-demand solution, 648
 Operation, 97
 Operation list in the production order, 260
 Operative environment, 297
 Operator
 Arithmetic, 134
 Character string operator, 135
 Logical, 134
 Order
 Bill of materials, 73, 93, 224, 255
 BOM, 220, 596

Combination, 462
Routing, 96, 255
Type, 255
 Order Change Management (OCM), 73, 260,
 334, 376, 388, 459
 Order Engineering Workbench, 223
 Organizational area, 100, 323
 Original package, 428
 Overall change profile, 334
 Overall profile, 389, 392

P

Packet type, 425
 Paper, 449
 Pattern Matching System, 375
 PDR → Product Data Replication, 410
 Pfunction, 176
 Pipes, 449
 Planned configuration, 450
 Planned independent requirement, 294
 Planned order, 264, 295
 Planning, 73, 283
 Profile, 285
 Strategy, 25, 70, 332, 333, 334, 389
 Table, 285
 Variant, 299, 300, 330
 Plant Maintenance, 261, 279
 PLM extension, 414
 PLM-Extension, 415
 PLM WebUI, 245
 PME
 Product Modeling Environment (Java PME),
 404
 PMEVC, 142, 214
 PMS → Pattern Matching System, 375
 Portal solution, 602
 Postponement, 450
 PP-Read Master Data, 260
 Precondition, 121, 164, 586, 591
 Precondition and variant table, 166
 Pricing, 58, 59, 72, 262, 272
 Procedure, 194
 Problem-solving processes, 559
 Procedural approach, 44, 45
 Procedural modeling method, 348
 Procedural object dependencies, 126
 Procedure, 56, 122, 168, 189, 586, 591
 Evaluate, 130
 Procedure and class node, 184
 Process automation, 658
 Procurement type, 90
 Product, 33
 Configurable, 34, 621, 655
 Designer, 233
 Model, 654
 Property, 655
 Specifiable, 651
 Specification, 34, 651
 Specified, 651
 Structure, 232
 Values, 42
 Variant, 34, 653
 Variant structure, 230
 Product and process flexibility, 649
 Product configuration, 31, 34, 36, 39
 Automated, 35
 Procedure, 34
 Product costing with quantity structure, 199
 Product Data Replication (PDR), 410
 Delta filtering, 423, 435
 Package posting, 436
 Replication of a VC model, 427
 Sending the package, 435
 Production configuration, 340
 Production order, 259, 264, 388
 Production order change management, 376
 Production rule system, 46
 Production scenario, 39
 Assemble-to-order, 41
 Configure-to-order, 41
 Engineer-to-order, 41
 Make-to-order, 40, 650
 Make-to-specification, 653
 Make-to-stock, 39
 Production tolerances, 451
 Product model
 Create, 69
 Product Modeling Environment (PME), 59
 Java PME, 59
 Product property, 655
 Product property format
 Boolean value, 655

Code, 655
Decimal number, 656
Free text, 656
Integer, 656
Quantity, 655
 Product specification, 34
 Product Structure Management (PSM), 248
 Product variant, 40
 Project BOM, 220
 Project Builder, 276
 Project system, 260, 262, 275
 Property
 Default value, 656
 Punch-out approach, 631
 Purchase order, 262, 272, 274
 Purchase requisition, 272

Q

QM blocked stock, 271, 274
 Quality Management, 261, 265

R

Read PP Master Data, 389
 Reassignment, 460
 Reconciliation Workbench (RWB), 251
 Reference characteristic, 58, 80, 189, 217, 591
 Access, 80
 Release key, 379, 390
 Replication table, 425, 439, 443
 Replication workbench, 427
 Report
 RUPSHIELEV, 423
 RUPSPPOST, 422, 438
 RUPSSSEND, 422, 435
 Required characteristic, 78
 Dynamic, 167
 Requirement
 Class, 329, 389
 Planning, 255
 Type, 258, 329
 Requirements planning, 264
 Requirements type, 275
 Resetting values, 45

Restrictable characteristics, 146, 150
 Restriction, 160, 161
 Restrict values, 353
 Routing, 39, 73, 377
 Simulatively exploded routing as a template, 205
 Running dot notation, 408
 Runtime version, 61, 213, 341, 373, 404

S

SaaS → Software-as-a-Service, 648
 Sales characteristics, 598
 Sales configuration, 340
 Sales Configuration Engine (SCE), 59, 375
 Advanced mode, 403
 LO-VC-compatible mode, 60
 Sdvanced mode, 61
 Sales order, 262, 388
 Bill of materials, 220, 225, 591
 Change management, 459
 Production, 450
 Routing, 225, 229
 Stock segment, 256, 257
 Sales Pricing Engine (SPE), 59
 Sales types, 614
 Sample model, 566
 SAP APO, 451
 SAP Apparel and Footwear Solution, 563
 SAP Business ByDesign, 32, 647
 SAP CRM, 48, 60
 CRM 5.0, 62
 SAP Custom Development, 224, 229, 576
 SAP Engineering & Construction, 449
 SAP enterprise solutions, 647
 SAP ERP, 32, 370
 ECC 6.0, 62
 ERP Central Component (R/3), 48
 SAP for Aerospace and Defense, 448
 SAP for Automotive, 449
 SAP for High Tech, 448
 SAP for Industrial Machinery & Components, 448
 SAP for Mill Products, 449
 SAP NetWeaver BW, 284, 286
 SAP NetWeaver technology platform, 62

- SAP Note
 - 68033, 296
 - 173756, 285
 - 174758, 285
 - 844816, 338
 - 844817, 316
 - 854170, 316
 - 901689, 375
 - 917987, 373
 - 997111, 375
 - 1081650, 373
 - 1121318, 375
 - SAP philosophy, 553
 - SAP PLM, 563
 - SAP Vehicle Management System, 563
 - Scenario
 - Order bill of material*, 107
 - Planned/production order with BOM explosion*, 118
 - Planned/production order without BOM explosion*, 105
 - Sales order (SET)*, 113
 - SCE → Sales Configuration Engine, 375
 - Schedule line category, 257, 331
 - SCREEN_DEP, 172
 - SDCOM-VKOND, 193
 - SD document categories, 254
 - Selection condition, 122, 167, 586, 591
 - Evaluate*, 130
 - Semi-declarative object dependencies, 126
 - Sequence
 - Dynamic*, 95
 - Sequences, 96
 - Serial number, 264, 271, 279
 - Profile*, 264
 - Service, 274
 - Material*, 283
 - Order*, 281
 - Service-oriented architecture (SOA), 648
 - Service product, 338
 - Simple BOM, 220
 - Simulative environment, 296
 - Single-level BOM explosion, 605
 - SKEY, 131
 - SOA → Service-oriented architecture, 648
 - Software-as-a-Service (SaaS), 528, 648
 - Solution for medium-sized businesses, 647
 - Specifiable, 349
 - Specified, 135, 348
 - SPE → Sales Pricing Engine, 59
 - Standard network, 48, 278
 - Standard product, 34
 - Planning*, 292
 - Standard work breakdown structure, 277
 - Start logo, 100
 - Strategy, 91, 257, 275
 - Group*, 257, 275
 - Planning strategy*, 330
 - Structure node, 230
 - Super BOM, 38
 - Supply Chain Management (SCM), 304
 - Switch framework, 447
 - Synchronization unit (GSS), 249
 - Syntax element
 - \$count_part*, 136
 - \$del_default*, 136, 169
 - \$PARENT.*, 133
 - \$part_of*, 136
 - \$ROOT.*, 133
 - \$SELF.*, 133
 - \$set_default*, 136, 169
 - \$set_pricing_factor*, 136, 195, 198
 - \$subpart_of*, 136
 - \$sum_part*, 136
 - Function*, 176
 - Inv*, 173
 - Invisible*, 172
 - Pfunction*, 176
 - Syntax rule, 132
 - System configuration, 60, 394, 395
- ## T
-
- Tab
 - Additional Data*, 80, 82
 - Basic data*, 77
 - Configuration Initial Screen*, 100
 - Configuration Parameters*, 101, 115
 - Descriptions*, 79
 - Fast Data Entry*, 454
 - Order BOM*, 111
 - Restrictions*, 80
 - Sales Order*, 115

- User Interface*, 102
- Values*, 79
- Table, 137, 138
 - Constraint*, 323
 - Constraint wizard in PMEVC*, 149
 - Formula*, 348
- Target system, 430
- Task list type, 95
- Technical characteristic, 598
- Test Case Editor, 521
- Test-Driven Development, 519
- Test-Driven Modeling, 521
- Tolerances, 268
- Tools, 75
- Trace, 140
- Trace function, 374
- Transaction
 - BD87 - IDoc Overview*, 435
 - BF01 - BTE Administration*, 417
 - C223 - Change Production Versions*, 560
 - CA75 - Replace Production Resources and Tools*, 560
 - CA85 - Replace Work Centers*, 560
 - CA95 - Change Reference Operation Sets*, 560
 - CAVC_TEST*, 669
 - CC01 - Create Change Master*, 378
 - CC03 - Display Change Master*, 385
 - CEWB - Engineering Workbench*, 560
 - CFM1 - Create Integration Model*, 306
 - CFM2 - Activate Integration Model*, 307
 - CL02 - Class*, 52
 - CL20N - Assignment of Objects to Classes*, 321
 - CL24N - Assignment of Objects to a Class*, 321
 - CLGT - Set Up Tables for Search*, 323
 - CLMM - Change Classification*, 560
 - CMOD - User Exits*, 317
 - CN08 - Process Network Parameter from Sales Order*, 278
 - COCM1 - Procurement Elements Initiating Object Records*, 391
 - COCM - Initiating Object Records*, 391
 - CRWBD - Replication Workbench*, 427
 - CS20 - Change BOM Data*, 560
 - CS40 Assignment of Configured Material*, 204
 - CS40 - Assignment of Configured Material*, 301
 - CS62*, 591
 - CSKB - Order Browser*, 223
 - CT04 - Characteristics*, 51
 - CT12 - Where-Used List for Characteristics/Characteristic Values*, 377, 387
 - CTBW - Table Maintenance for BW and Classes*, 286
 - CU05 - Dependency Where-Used List*, 387
 - CU50 - Configuration Simulation*, 388
 - CU51E - Result-oriented order BOM*, 581
 - CU51 - Order BOM*, 223
 - CU60E*, 154
 - CU60E - Upload Spreadsheet to Variant Table*, 629
 - CU61 - Create Variant table*, 372
 - CU62 - Change Variant Table*, 372
 - CUMODEL*, 138
 - EXPO_TEST - Test Structure Explosion by FOX*, 442
 - IP10 - Schedule Maintenance Plan*, 280
 - IP41 - Create Maintenance Plan*, 280
 - MC(B - Standard Analysis Variant Configuration*, 284
 - MCSZ - Copy Management*, 284
 - MD04 - Stock/Requirements List*, 272
 - MD04 - Stock/Requirements List*, 264
 - MD50 - Sales Order Planning*, 264
 - MDP1 - Create Planning Table*, 293
 - MDPH - Planning Profile*, 293
 - MDPV - Type Matching*, 301
 - MIGO - Goods Movement*, 271
 - MM01 - Create Material*, 49
 - MM17 - Change Material Master Data*, 560
 - MM50 - Extend Materials*, 560
 - MS02 - Long-Term Planning*, 299
 - MS31 - Create Long-Term Planning Scenario*, 299
 - MS66 - Copy Simulative Dependent Requirements*, 297
 - NWBC - NetWeaver Business Client*, 245
 - /OEWB/MAIN - Order Engineering Workbench*, 224

OISD - Service Products, 283
 OMO1 - Activate Update, 285
 OVRP - Update Group Change Item Level, 285
 PCFG - Role Maintenance, 317
 PMEVC - Modeling Environment, 57
 PMEVC - Modeling Environment Variant Configuration, 316
 PMEVC - Modeling Environment for Variant Configuration, 629
 PMEVC - Product Modeling Environment for Variant Configuration, 53
 PPECS - BOM Converter, 242
 QE51N - Results Recording QM, 267
 RSA2 - SAPI DataSource Repository, 287
 SCU0 - Customizing Cross-System Viewer, 442
 SLG1 - Application log, 439
 SLG1 - Application Log, 442
 SM37 - Job Selection, 432, 434
 SM50 - Process Overview, 434
 ST01 - Authorization Trace, 442
 ST05 - Performance Analysis, 375
 STAD - Business Transaction Analysis, 375
 SU03 - Maintain Authorizations, 404
 UPS03 - Uniform Packaging Service, 628
 UPS - UPS Cockpit (PDR), 436
 UPSRCP - Post UPSRCP IDocs, 443
 UPSSETUP - Customizing Preparation for PDR, 413, 417
 VA01 - Create Sales Order, 370
 WE20 - Partner Profiles, 419
 Transaction Tax Engine (TTE), 59
 TTE → Transaction Tax Engine, 59
 Type matching, 301
 type_of, 135
 Type of building block, 405
 Types, 612

U

UPSMAS, 413
 UPSRCP, 413
 UPS (Uniform Packaging Service), 421
 User exit, 564
 User interface design, 177

V

Valuation for a single batch, 460
 Value assignment type, 78
 Value set, 35
 Variability, 582
 Variable in constraints, 160
 Variant, 34
 Bill of materials, 57, 92, 221
 Class, 49, 52, 72, 82, 377
 Class and PMEVC, 146
 Class type, 321
 Condition key, 58
 Condition record, 194
 Conditions, 586
 Configurator LO-VC, 48, 55, 647
 Diversity, 41
 Function, 56, 137, 174
 Model browser, 138
 Parts, 182
 Routing, 57
 Table, 56, 136, 146, 372, 378, 586, 591, 657
 Table and procedure, 170
 Variant Configuration, 35
 Basic principles, 31
 Integration, 69
 Main tasks, 58
 Modeling, 69
 Variant diversity, 41
 Variant planning, 300
 Variants, 35
 Class, 400
 LO-VC configuration, 400
 LO-VC Variant Configuration, 397
 VCSD_UPDATE, 171
 Versioning, 227, 228
 Versioning of order BOMs, 226
 View node, 231
 Virtual Machine Container (VMC), 62
 VMC → Virtual Machine Container, 62

W

Web channel, 337
 WF-BATCH (workflow user), 434

Where-used list for characteristics/
characteristic values, 86
Work breakdown structure, 276
Workflow Customizing, 426
Workflow system, 591

X

XCM → Extended Configuration Management,
608
XCM scenario, 316