Specification

OCD
OFML Commercial Data*
(OFML Part IV)

Version 4.3

Status: Release

Thomas Gerth, EasternGraphics GmbH (Editor)

2020-06-25

Contents

1 Introduction

OCD principally serves to create product data which is needed and exchanged within business processes of the furniture trade. Primarily, OCD is supposed to cover and process the following tasks:

- Configuration of complex articles
- Price determination
- Creation of offer and order forms

OCD is *not* a format for catalog data creation. The catalog data must be provided otherwise. The link between catalog and product data is carried out by the respective software system based on the the article numbers.

The data modell for OCD is based on the fundamental OFML product data model (see appendix A of the OFML standard, version 2.0.2).

CSV tables (comma separated values) are used as physical exchange format between OFML conform applications. The following regulations apply for this:

- Each of the below described tables is included in exactly one file. The file name is made of the prefix "`ocd_`", the specified table name and the suffix "`.csv`"; the table name is written completely in small letters.
- Each line of the file (ended by a character for the line wrap '`\n`') represents a data record. Blank lines (consisting of zero or several blank characters or tabulator) are ignored. `ISO-8859-1` (Latin-1) is used as the character set.
- The fields of a data record are separated from each other by a semicolon.
- Lines starting with a number sign ('`#`'), are interpreted as a comment and excluded from the further processing.

In the following table descriptions, a field of a data record is specified by the following attributes:

- Number
- Name
- Mark, whether the field belongs to the primary key of the table
- Data type (see below)
- Maximum length of the field (number of characters)[1]
- Mark, whether the field has to be filled (obligatory field)

[1] While in principle there are no restrictions in CSV data records concerning individual field lengths, for certain fields of data type `Char` maximum possible resp. reasonable lengths resulting from the intended purpose are specified here. Moreover, in data creation further restrictions that are imposed by the program used in the data creation process should be observed.

The following **data types** are defined:

Char Character string

The following lexical and syntactical regulations are valid:

1. All printable characters, except the field separation character (semicolon), are allowed.
2. If a semicolon is meant to be included in the character string, the whole field has to be enclosed in quotation marks (’"’) (which are not followed or preceded by a single quotation mark). The opening and the ending quotation mark are not taken over, when the field is read.
3. If the field is enclosed in quotation marks, two successive quotation marks are replaced by a single quotation mark, when the field is read. A single quotation mark in a field enclosed in quotation marks is not allowed.
4. If the field is enclosed in quotation marks, blank characters between the ending quotation mark and the next field separation character or the line end are ignored.

Num Number

all numbers (may include decimal point and possibly a minus sign at the first position)

Bool Boolean value

’1’ – yes, ’0’ – no

Date Date

Format: 4 digits year + 2 digits month + 2 digits day (corresponds to ISO 8601, hyphens between year, month and day are omitted)

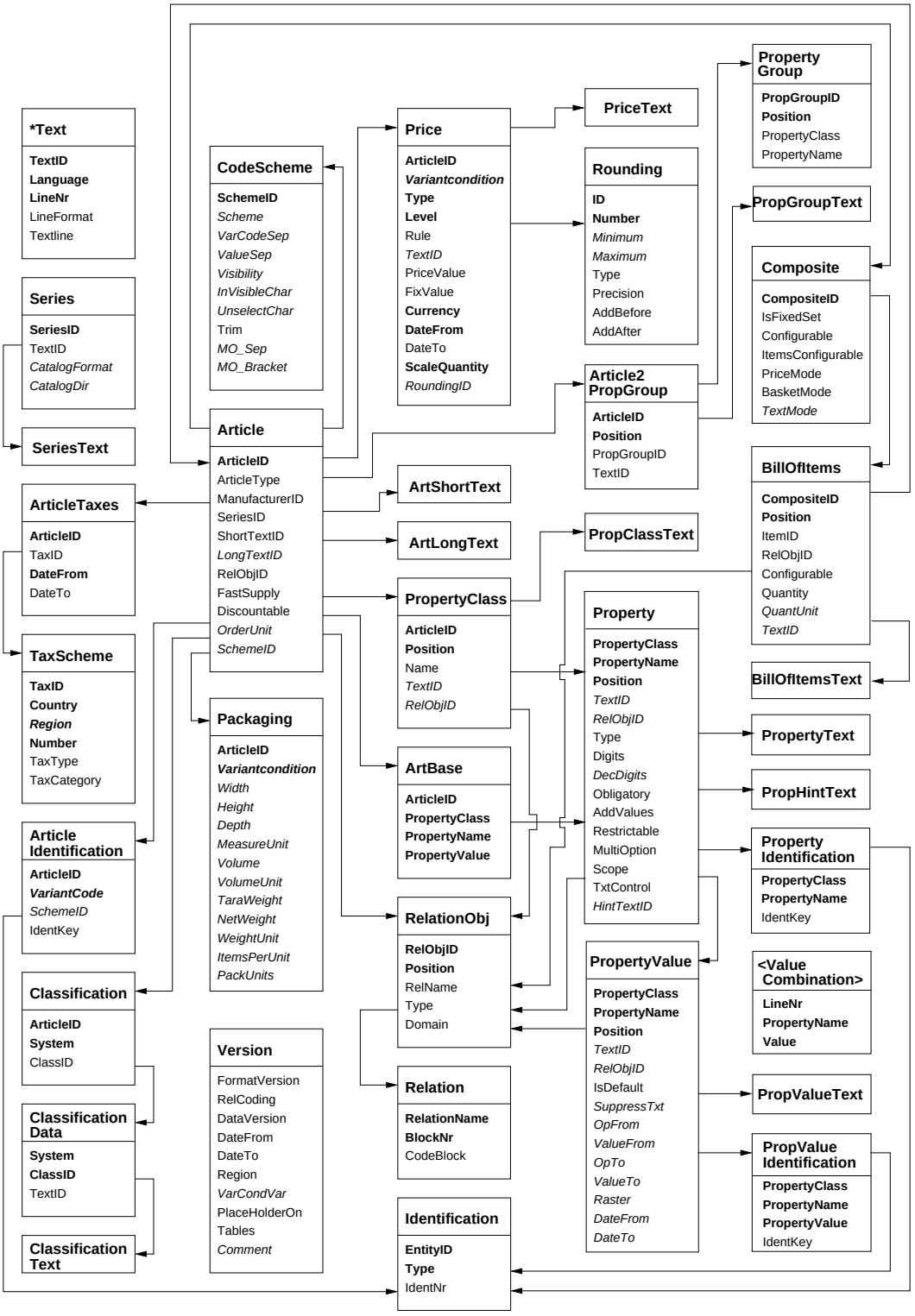
The obligatory field mark is only relevant for fields of the data type *Char*. The fields of the other data types always have to be filled. Concerning the fields of the data type *Num*, the respectively possible values can be found in the description of the respective tables.

Fields for *units* are defined in several tables. The indication of units in OCD follows the standard `openTRANS` for the inter-industry electronic exchange of business documents. According to this, units are indicated according to the *Common Code* of the UN/ECE Recommendation 20[2].

[2] www.unece.org/cefact/rec/rec20en.htm

2 The tables

2.1 Overview



Primary key fields are highlighted by bold print and the fields, which are no obligatory fields, by italic print. The structure of the text tables is only shown once. The alternative text tables (see section 2.20) are not presented for the sake of clarity.

2.2 The article table

Table name: `Article`
Obligatory table: yes

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	ArticleID	X	Char		X	Base article number
2.	ArticleType		Char		X	Article type
3.	ManufacturerID		Char		X	ID of manufacturer
4.	SeriesID		Char		X	ID of series
5.	ShortTextID		Char		X	Short text number
6.	LongTextID		Char			Long text number
7.	RelObjID		Num		X	Relational object number
8.	FastSupply		Num		X	Fast supply counter
9.	Discountable		Bool		X	discountable article?
10.	OrderUnit		Char	3		Order unit
11.	SchemeID		Char			Identifier of the codification scheme for the generation of the final article number

Remarks:

- This is the main table for all articles. The article number is used as key to access further tables for the price determination, classification, etc.[3].
  The base article number (model number) used by the manufacturer has to be indicated here as article number. Further numbers concerning the identification of the article in different contexts can be entered via the article identification table (see section 2.3).

- The article type (field 2) determines fundamental attributes of the article. The following article types are possible:

Article type	Explanation
P	plain article: not configurable, no sub items
C	configurable article: Properties of the article can be set by the user, no sub items
CS	composite articles: can include sub items, can be configurable itself

- The manufacturer ID's (field 3) will be assigned and managed centrally by the company Eastern-Graphics GmbH.

- For the series ID (field 4) only the upper case letters and digits from the ASCII character set as well as the underscore '_' should be used.
  The usage of the following characters is *not* recommended: `\/?:*"><|,;=` as well as the space character[4].

- The text numbers (fields 5 and 6) are used as key for the tables with the short resp. long descriptions of the articles (see section 2.20).

- The relational object number serves as key to access the relational object in the relational object table (see section 2.15), to which relationship knowledge concering the article is linked. (If no relational object is needed, the number `0` has to be indicated.)

[3] The allocation of the article-related information to different tables increases the clarity by hiding optional information and facilitates the extensibility as well as the incremental data exchange.

[4] If these characters are used, correct data processing on all platforms and in all application systems is *not* guaranteed.

- The fast delivery counter (field 8) determines the number of the articles from which onwards a fast delivery is possible. The number `0` indicates that a fast delivery for this article is generally not possible.
- The flag in field 9 specifies whether discounts can be applied to the possibly specified purchase price of the article or not. A value of `0` (false) means, that, deviating from the general conditioning of the producer, *no* deductions from the purchase price are allowed for this article.
- for the product Field 10 specifies the unit in which the article can be ordered. Both the quantity indication in an order and the price in the price table refer to this unit.
  The unit has to be indicated according to the Common Code of the UN/ECE Recommendation 20. Common units for the furniture industry are `C62` – *piece*, `MTR` – *meter* and `MTK` – *square meter*. If no indication is given, *piece* is used as default unit.
- The identifier indicated in the 11th field serves to reference the codification scheme from the table `CodeScheme` (section 2.24), which is used for the generation of the final article number. If no identifier is indicated or an identifier, which is not referenced in the scheme table, no specific final article number will be generated for the article[5].

2.3 The article identification table

Table name: `ArticleIdentification`
Obligatory table: no

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	ArticleID	X	Char		X	(Base) article number
2.	VariantCode	X	Char			Code of the article variant
3.	SchemeID		Char			Scheme for variant code
4.	IdentKey		Char		X	Key for identification table

Remarks:

- The table serves to indicate additional identification numbers (of different types) for articles. The additional identification numbers are not directly stored in this table, but indirectly in the table `Identification` (see section 2.22) via the key indicated in field 4.
- If the identification number is not supposed to be assigned to the base article, but to a certain variant of the (configurable) article, the variant has to be specified in the 2nd field with the help of a corresponding code. The codification scheme used for this purpose has to be indicated in field 3 (key for coding scheme table, see section 2.24).
- If there are several entries for a base article in the table, a variant code has to be indicated for each entry. If no corresponding variant code is found to a given configuration of an article, no identification of the desired type can be determined for this article.

[5] The final article number is then equal to the basic article number.

2.4 The classification table

Table name: `Classification`
Obligatory table: no

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	ArticleID	X	Char		X	Base article number
2.	System	X	Char		X	Name of the classification system incl. version
3.	ClassID		Char		X	ID of the class of the article

Remarks:

- The table serves to classify an article.
- An article can be classified according to different classification systems. Currently, the following systems are allowed:

System	Explanation
ECLASS-x.y	Classification according to the eClass model with indication of the version
UNSPSC	Classification according to the standard UN/SPSC
<Manufacturer>_*	Manufacturer-specific classification: the system designation is built out of the manufacturer ID, an underscore and an arbitrary addition

  Manufacturer-specific classifications can be used to define product groups, product hierarchies and others.

Table name: `ClassificationData`
Obligatory table: no

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	System	X	Char		X	Name of the classification system incl. version
2.	ClassID	X	Char		X	ID of the class
3.	TextID		Char		X	Text number

Remarks:

- The table serves to indicate information for a classification[6].
- The text number serves as key for the table `ClassificationText` (see section 2.20), in which language-specific texts describing the classification can be stored.

[6] Presently, solely the indication of a text to describe the classification is supported.

2.5 The packaging table

Table name: `Packaging`
Obligatory table: no

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	ArticleID	X	Char		X	Base article number
2.	Variantcondition	X	Char			Variant condition
3.	Width		Char			Width of the packaging unit
4.	Height		Char			Height of the packaging unit
5.	Depth		Char			Depth of the packaging unit
6.	MeasureUnit		Char	3		Unit of measurement of the dimensions 3 to 5
7.	Volume		Char			Volume of the packaging unit
8.	VolumeUnit		Char	3		Unit of measurement of the volume
9.	TaraWeight		Char			Weight of the packaging unit
10.	NetWeight		Char			Weight of the individual article
11.	WeightUnit		Char	3		Unit of measurement of the weights 9 to 10
12.	ItemsPerUnit		Char			Number of the articles per packaging unit
13.	PackUnits		Char			Number of the packaging units, which are used for the article

Remarks:

- This table serves to indicate information conerning the packaging of an article which is delivered completely.
- A packaging unit can include several articles (of the same number)(field 12). Otherwise, components of the article, e.g. optional accessories, could be delivered in separate packaging units (field 13).
- Dimensions, volumes, weights and numbers of packaging units may deviate from the basic condigiuration of the article, depending on special properties. Such situations can be represented with the help of *variant conditions* (field 2). The usage and handling of such variant conditions is described in section 6. The amounts of the entries with variant conditions (non-empty field 2) are always indicated as difference to the respective basic amount of the entry without variant condition and can be negative[7].
- If field 2 (variant condition) is not empty, in field 1 the wildcard character "*" can be used to indicate uniform values for variant-specific dimensions, volumes and weights, which are valid for various articles.
  However, this article-independent table entry is taken into account only if there is no own, specific entry with the same variant condition for the processed article.
- Fields 3-5, 7, 9, 10, 12 and 13 are declared as optional string fields, so they may be empty. If they are not empty, the fields have to contain string representations of numeric values[8].

[7] This implies that for a given article, a table entry without variant condition must always be present. The data elements in it can have a value of `0.0` if necessary.

[8] corresponding to the representation of numeric values in the value table

- The following units of measurement are allowed in the fields 6, 8 and 11[9]:

  Dimensions (field 6):

Code of unit of measurement	Explanation
CMT	Centimeter
FOT	Foot
INH	Inch
MMT	Millimeter
MTR	Meter

  The field must not be blank if one of the fields 3-5 is not empty.

  Volume (field 8):

Code of the unit of measurement	Explanation
INQ	Cubic inch
LTR	Liter
MTQ	Cubic meter

  The field must not be blank if field 7 is not empty.

  Weight (field 11):

Code of the unit of measurement	Explanation
KGM	Kilogram
LBR	Pound
MGM	Milligram

  The field must not be blank if one of the fields 9 and 10 is not empty.

- The volume can differ from the volume calculated from width, hight and depth, if packaging is used that can be stacked into one another.
- The total weight (gross) of a packaging unit results from the weight of the packaging (field 9) plus the product of the weight of the individual article (field 10) and the number of articles per packaging unit (field 12).

[9] corresponds to the Common Code of the UN/ECE Recommendation 20 (www.unece.org/cefact/rec/rec20en.htm)

2.6 The table of composite articles

Table name: `Composite`
Obligatory table: no[10]

Composite articles are articles which include a fixed or variable number of sub items. This can be a simple aggregation in the sense of a set, but also a composition from functional aspects.

No.	Name	Key	Type	L"ength	Oblig.	Explanation
1.	CompositeID	X	Char		X	Article number of the composite article
2.	IsFixedSet		Bool		X	Number of the sub items fixed ?
3.	Configurable		Bool		X	Composite article configurable ?
4.	ItemsConfigurable		Bool		X	Sub item configurable ?
5.	PriceMode		Char	3	X	Mode of the price determination
6.	BasketMode		Char	1	X	Mode of the presentation of the sub items in the basket
7.	TextMode		Char			Mode of the text presentation of the sub items in the basket

Remarks:

- The table serves to determine the general attributes of a composite article. The sub items are determined in the bill of items (see next section).
- Field 2 indicates, whether the number of the sub items is fixed. If *no* is indicated, the number of the sub items varies depending on the existence conditions to be indicated for the sub items in the bill of items.
  If *yes* is indicated, possibly indicated existence conditions for sub items will not be evaluated!
- Field 3 indicates, whether the composite article is configurable itself. If *no* is indicated, the composite article cannot be configured, even if property classes are assigned to it (see section 2.8).
- If the value in field 4 is *no*, the sub items are generally not allowed to be configured, even if this is otherwise specified for individual sub items in the bill of items.
  On the other hand, individual sub items in the bill of items can be excluded from the configurability, even if the configurability is generally allowed through value *yes* in field 4.
- The mode in field 5 derternines the manner of the price determination of the composite article:

Price mode	Explanation
C	Price is bound to the composite article (with variant conditions, if applicable)
S	Price results from the sum of the prices of the sub items
C+S	Price of the composite article plus sum of the prices of the sub items

  Principally, each of the three price modes is imaginable/possible for each possible value combination of the fields 2-4. There is no restriction. The coherence has to be observed/guaranteed during the data recording. If the sub items are configurable for example (field 4 true), the price mode "C" usually does not make sense, unless it is guranteed by relationship knowledge (see below) that the value sets of all price-relevant properties of the sub items are restricted to exactly one value.
- The presentation of the sub items in commercial forms can be controlled via the mode in field 6:

[10] The table is only needed, if the database is indeed meant to include composite articles.

Basket mode	Explanation
H	Sub items are presented as sub positions (i.e. hierarchically) without price, if applicable
T	the sub items are listed in the description text of the composite article

- The mode in field 7 indicates how the sub items in basket mode "T" are supposed to be described in the description text of the composite article[11]:

Text mode	Explanation
BAN	via base article number
FAN	via final article number
ST	via article short text
LT	via article long text
BAN+ST	via base article number and article short text
BAN+LT	via base article number and article long text
FAN+ST	via final article number and article short text
FAN+LT	via final article number and article long text
ST+BAN	via article short text and base article number
ST+FAN	via article short text and final article number
LT+BAN	via article long text and base article number
LT+FAN	via article long text and final article number

2.7 The bill of items

Table name: `BillOfItems`
Obligatory table: no

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	CompositeID	X	Char		X	Article number of the composite article
2.	Position	X	Num		X	Position of the sub item
3.	ItemID		Char		X	Article number of the sub item
4.	RelObjID		Num		X	Relational object number
5.	Configurable		Bool		X	Sub item configurable ?
6.	Quantity		Num		X	Quantity
7.	QuantUnit		Char	3		Quantity unit
8.	TextID		Char			Text number

[11] This mode is not necessary for the basket mode "H", because the presentation of the sub items as order position follows the standards of the respective application.

Remarks:

- This table indicates, which sub items can be included in a composite article (see previous section).
- The position of a sub item within the composite article (field 2) is taken into account in order listings.
- An existence condition for the sub item can be indicated via relational object specified in field 4 (see section 2.15). Existence conditions have to be indicated as relation type *precondition* with the scope "BOI". If an existence condition is specified and not fulfilled, the sub item will not be included in the current bill of items of the composite article.
  However, existence conditions in general are only evaluated, if *no* is indicated for the composite article in the field *IsFixedSet* of table `Composite` (see previous section)
- Field 5 indicates, whether the sub item is configurable. If *no* is indicated, it cannot be configured, even if property classes (see section 2.8) are assigned to it. However, sub items in general are only configurable, if *yes* is indicated for the composite article in the field *ItemsConfigurable* of table `Composite`.
- Via the quantity specification in field 6, identical articles can be summarised at a BOM list position. The sub items associated with this position then must not be configured, i.e. the value in field 5 is ignored[12].
- Field 7 specifies the unit to which the quantity specification in field 6 refers. The unit must be specified in accordance with the Common Code of UN/ECE Recommendation 20. If there is no specification, *piece* (`C62`) is used as the standard unit.
- The text number in field 8 is used as the key for the table `BillOfItemsText`. In this table texts can be stored which have to be represented in commercial forms for a BOM position additional to standard article information. The manner of this representation depends on the used OFML application.

2.8 The property class table

Table name: `PropertyClass`
Obligatory: yes[13]

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	ArticleID	X	Char		X	Article number
2.	Position	X	Num		X	Position of the class
3.	Name		Char		X	Identifier of the class
4.	TextID		Char			Text number
5.	RelObjID		Num		X	Relational object number

[12] Note: If the sub items also have to be graphically represented in an OFML–based application, before OCD data creation it must be clarified whether the particular application supports the graphical representation of summarised BOM items.
[13] The table can be omitted, if the database is not meant to store configuration data, but e.g. only article texts and prices.

Remarks:

- In this table, the property classes, which describe the properties of the article, are assigned to the articles[14].
- The position of the class (field 2) within the set of the property classes of an article influences the order in listings of the properties of the article (variant texts, property editors, and others).
- For the identifier of a property class (field 3), all alphanumeric characters including the underscore are allowed, but the first character must not be numeric.
- Relations of type *precondition* and *action* can be bound to the property class via the relational object specified in field 5 (see section 2.15). (If no relational object is needed, the number `0` has to be indicated.)

2.9 The property table

Table name: `Property`
Obligatory table: yes[15]

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	PropertyClass	X	Char		X	Identifier of the property class
2.	PropertyName	X	Char		X	Identifier of the property
3.	Position		Num		X	Position of the property
4.	TextID		Char			Text number
5.	RelObjID		Num		X	Relational object number
6.	Type		Char	1	X	Data type of the property values: C - Char T - Text N - Num L - Length
7.	Digits		Num		X	Number of the digits (total)
8.	DecDigits		Num			(thereof) number of the decimal digits
9.	Obligatory		Bool	1	X	Is input mandatory?
10.	AddValues		Bool	1	X	Additional values allowed?
11.	Restrictable		Bool	1	X	Value set restrictable?
12.	MultiOption		Bool	1	X	Multivalued property?
13.	Scope		Char	2	X	Scope, see table below
14.	TxtControl		Num		X	Text control
15.	HintTextID		Char			Text number for hint text

Scope	Explanation (details see below)
C	configurable (visible)
R	only in relational knowledge
RV	not configurable, but visible for users
RG	not configurable, but graphic-relevant

[14] The term *property* is used throughout this specification as a synonym for *feature* or *characteristic*.
[15] The table can be omitted, if the database is not meant to store configuration data, but e.g. only article texts and prices.

Remarks:

- In this table, the properties of each property class are listed.
- The names of the properties (field 2) are symbolic (language independent) identifiers.
  All alphanumeric characters can be used, including the underscore, but the first character must not be a numeric character.
  Descriptive (language dependent) names (for the use in the user interfaces) are stored in the table `PropertyText` (see section 2.20). For this purpose, a text number is assigned as access key in the 4th field.
  The indication of a text number is only necessary for visible properties according to field 13 (see also remark below).
- Within the property classes of an article a given property (field 2) may occur only once[16].
- The position of the property (field 3) within the property class influences the order in listings of the properties of the article (variant texts, property editors, etc.).
- The relational object number (field 5) serves as key to access the relational object in the table `RelationObj` (see section 2.15), to which the relational knowledge for the property is linked. (If no relational object is necessary, the number `0` has to be indicated.)
- The data type (field 6) defines the type of the values which can be assigned to the property and determines the manner of the presentation of values of the property in the property value table:
  - Values of the data type 'C' are simple character strings with a maximum length according to the indication in field 7. All representable characters from the character set `ISO-8859-1` (Latin-1) can be used, except the space and the backslash ('\').
  - The data type 'T' is used for properties, where the user can enter any text including line breaks. Values of this type are strings with a maximum length as indicated in field 7. (If field 7 contains the value `0` there is no length limit). In the value string line breaks are represented by a newline character ('`\n`'). In the property value table no values should be stored (resp. they will be ignored).
  - Values of the data types 'N' and 'L' are real or integral numbers, where real numbers are represented in a simple decimal point notation. Field 7 determines the maximum number of all digits without decimal point, and field 8 determines the number of digits used thereof for the presentation of the fractional part. With negative numbers, the minus sign is at the first position (so for the digit positions, there is one fewer digit available than specified in Field 7).
  - Essentially, the data types 'N' and 'L' (field 6) are handled in the same way. The difference lies in the format indication of the OFML property, which is generated for the respective OCD property (see also OFML documentation, section "Format specifications for properties" in the appendix "Format specifications"):
    - The format for properties of the type 'N' is `%<Field7>.<Field8>f`, if the number of the decimal digits (field 8) is unequal 0, otherwise `%<Field7>d`.
    - The format for properties of the type 'L' is `@L`, which requires the property editor of the application to represent resp. enter the value of the property using the unit of measurement set by the user. The property editor therefore performs a conversion between the user-defined unit of measurement and the unit of measurement (m) used in OFML for dimensions. I.e., values for properties of this type in the table `PropertyValue` have to be indicated in meters.
- Field 9 specifies, whether the property is an obligatory property (1) or an optional property (0). An obligatory property has to have a value assigned by the user. As long as an article has obligatory properties with no assigned value, its configuration is not complete (invalid).
  This attribute is only relevant for properties of the type 'C'.
  If a value set (see table `PropertyValue`, section 2.13) is specified for a property marked as optional,

[16] Otherwise relationships (see section 2.15) can not be clearly evaluated, since there the properties are not qualified by their class.

the application system automatically generates and uses additionally the pseudo value `VOID` for the state "not selected" ("not assigned").
For obligatory properties with a specified set of property values and without the possibility of a free value input (see field 10), the application system guarantees that a value from the value table is always selected; such a property thus always has an assigned value. For other properties, the completeness of the configuration has to be assured by providing according product relationship knowledge (selection conditions, see table `RelationObj`, section2.15).
Note: this attribute refers to the input obligation of the user. There is no direct connection with the product-logical term *Option* for properties, where the user can choose – but does not have to – from a set of values (choice list). Options from a product-logical point of view can also be represented via obligatory OCD properties. With numeric properties this is even necessary (see above); the state "not selected" can and has to be represented by a special value (e.g. `0`).

- The attribute in field 10 (`AddValues`) indicates, whether the user can input values freely, may be in addtion to the values specified for the property in the value table. This can be used to enter free texts, quantities or measures.
  Note: If the user may input values for a numeric property only in certain allowed ranges, the according interval values have to be specified in the value table (see table `PropertyValue`, section 2.13) and the value of this attribute has to be `0` (*false*).

  This attribute is only relevant for simple (single-valued and non-restrictable) configurable properties, with the following restrictions:

  – To properties of type T as well as to properties, for which no values are specified in the value table, generally can be assigned any value, i.e. even if this attribute has value `0` (*false*).
  – Are there specified values for the property in the article base table, only these values can be assigned or set, i.e. this attribute then is ignored.

  For all other property types, this attribute is of no relevance. Instead, regarding the assignment of values the following provisions apply:

  – To non-configurable properties generally can be assigned any value, unless there are specified values for the property in the value table: then only these values can be assigned.
  – To multi-valued and restrictable configurable features can be assigned only the values specified for the property in the value table.

- If the value set of the property is supposed to be restrictable with the help of relations of the type *Constraint* (see section2.15), the attribute in field 11 (`Restrictable`) has to be set to `1` (*true*).
  With normal, *non*-restrictable properties, the value set (choice list) from which the user can choose, comprises all values which are specified for the property in the property value table **and** which either have no preconditions or whose preconditions are fulfilled with regard to the current configuration of the article. (Note: Obligatory properties have always assigned one of the values.)
  Restrictable properties are handled in a different way. The value set of these properties is restricted by contraints, starting with the full set of values specified in the property value table. Restrictable properties are considered to be evaluated only when the value set is either restricted to exactly one value via a constraint or when the user has made a selection. The configuration of an article is complete only when all restrictable properties are evaluated. If a restrictable property is not evaluated, the article cannot be ordered.

- Multi-valued properties (field 12) are properties, which can have several values at the same time (e.g. "special equipment").

- The *scope* (field 13) indicates, whether the property is allowed to be configured (modified) by the user of a configuration system, whether it is visible for the user and whether it is needed for the generation of the graphical representation of the article:

  – Only properties of scope "C" (or with blank field 13) are configurable. They are per se also visible and can be used for the generation of the graphical representation.
  – Properties of all other scopes are helping properties, which can be used within relationship knowledge.

  - Properties of scope "RV" are shown to the user as read-only and may also be used for the generation of the graphical representation.
  - Properties of scope "RG" are not visible for the user, but they may be used for the generation of the graphical representation.

  The scope also has influence on the persistence and initialization:
  - Properties of scope "R" are only used within the evaluation of relations. Therefore, the current state of these properties is not available outside of a configuration procedure (i.e., is not persistently stored with the article). A consequence of this is that properties of this scope are initialized at the beginning of each configuration procedure (see below).
  - The state of the properties of all other scopes is needed even after the termination of a configuration procedure[17] and therefore is persistently stored with the article. Thus, the initialization of these properties is performed only once, right during article creation.

- Field 14 specifies a code which controls the generation of the text describing the property in commercial forms (bill of items and others). The manner of the control is more fully described in section 5. (The code `0` identifies the standard procedure for single line texts.)

- In field 15, a text number (access key) can be specified, under which a (multi-line) hint text on the property is stored in the table `PropHintText`. This text can be displayed by an application as a hint[18] if the user moves the mouse pointer over the property identifier in the application's property editor.
  The following structure is recommended as the basic composition of a property hint:
  - Property identifier (as assurance in case part of the property identifier in the property editor is not visible due to the column formatting)
  - Information on valid values ranges
  - Other instructions for use

Between the attributes of a property there are *dependencies*, which limit possible combinations of attributes. These dependencies are described with the following rules:

- The flags *Obligatory* and *AddValues* are only relevant for configurable properties (Scope C).
- Configurable numeric properties are always mandatory properties[19].
- Configurable restrictable properties are always mandatory properties.
- Properties for free string or text input by the user are mandatory properties, i.e an empty string or text is a real value.
- Properties for free text input by the user (type T) implicitly are configurable (i.e. scope C is assumed). Indicators *Restrictable* and *MultiOption* are not relevant for these properties. Furthermore, only text control codes `0` and `5` are allowed (see appendix 5).
- Non-configurable properties can be restrictable. However, unlike configurable restrictable properties, these do not have to be evaluated (via relationship knowledge) in order to complete the configuration of the article.
- With (configurable) restrictable properties, it is not possible to input additional values.
- For configurable restrictable properties, there must be a value set specified or a constraint must exist (and be effective) that performs an evaluation of the property. Otherwise, the configuration would never be complete.
- Interval values are only relevant for configurable non-restrictable numeric properties.
- Type L is only useful for visible properties (Scopes C and RV).
- Properties of type T cannot be multi-valued.

[17] e.g. for display in property editors or for creation of the grahics of the article
[18] temporary text box at the location of the mouse pointer
[19] Numeric properties always require a value, because otherwise the system cannot execute any operations with such properties.

- For multi-valued properties, the flags *AddValues* and *Restrictable* are not relevant.
- Interval values are not possible for multi-valued properties.

The properties of the different types and scopes are initialized as follows[20]:

- Properties of type T will be initialized with an empty string.
- The initial state of restrictable properties is undefined (without value).
- Non-restrictable *configurable* properties are initialized with the value from the value table marked as the default[21].
  If there is no value in the value table marked as the default, mandatory properties are initialized with the first value from the set of values, whereas the initial state of optional properties then is undefined.
  If there are no values at all specified in the value table, properties of type C are initialized with an empty string and properties of the types N and L are initialized with the value `0` resp. `0.0`.
- Non-restrictable *non-configurable* properties are initialized with the first value from the article base table, but not properties of the scopes RG and RV, if there are values specified for these properties in the value table.
  If there are no values specified in the article base table, but in the value table, the behavior is as for configurable properties. (However, for properties of scope R values from the value table only will be considered if there is specified exactly one value or if the values are provided with preconditions.)
  If there are specified values neither in the article base table nor in the value table, the initial state of properties of scope R is undefined, whereas properties of the scopes RG and RV depending on the type are initialized with an empty string or `0` resp. `0.0`.

2.10 The property identification table

Table name: `PropertyIdentification`
Obligatory table: no

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	PropertyClass	X	Char		X	Identifier of the property class
2.	PropertyName	X	Char		X	Identifier of the property
3.	IdentKey		Char		X	Key for the identification table

Remarks:

- The table serves to indicate additional identification numbers (of different types) for properties.
- The additional identification numbers are not directly stored in this table, but indirectly in the table `Identification` (section 2.22) via the key indicated in field 3.

[20] Initialization means the value assignment of a property prior to the evaluation of relations. Afterwards, in actions and constraints the value still can be changed.

[21] Only one value may be marked as the default. If there are notwithstanding several values marked as the default, the behavior of the application is undefined.

2.11 Property groups

Property groups can be used by OFML applications to group the properties of an article in the property editor. For that purpose, the following two tables have to be provided:

Table name: `Article2PropGroup`
Obligatory table: no

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	ArticleID	X	Char		X	Base article number
2.	Position	X	Int		X	Position of the property group
3.	PropGroupID		Char		X	ID of the property group
4.	TextID		Char			Text number

Remarks:

- In this table, one or more property groups will be assigned to an article.
  If the table does not contains entries for a given article, an OFML application may group its properties based on its property classes.
- The position (field 2) determines the order of the property groups when displayed in the property editor.
  If there only one property group is specified for an article, and if this group contains all (currently visible) properties of the article, the application may renounce the display of the group (flat style).
- The ID of the property group (field 3) is used as a foreign key for table `PropertyGroup` (see below).
- The text number (field 4) is used as a foreign key for table `PropGroupText` (see section 2.20), in order to specify a language dependent name for the property group.
  In principle, different text numbers, and thus different names, can be specified for one and the same property group with different articles. However, normally the same text number for a common name is used.
  If there is no language dependent name specified for a given language and a given group, the ID of the group is used for display in the property editor.

Table name: `PropertyGroup`
Obligatory table: no

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	PropGroupID	X	Char		X	ID of the property group
2.	Position	X	Int		X	Position of the property
3.	PropertyClass		Symbol		X	Identifier of the property class
4.	PropertyName		Symbol		X	Identifier of the property

Remarks:

- This table defines the properties of a property group.
- The position (field 2) determines the order of the properties within the property groups when displayed in the property editor.
- Properties (fields 3 and 4) not possessed by a given article are ignored.
- Currently visible properties of the article, which are not listed in the property groups of the article, appear after these groups in an artificial property group ”Other” in an undefined order.

2.12 The article base table

Table name: `ArtBase`
Obligatory table: no

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	ArticleID	X	Char		X	Article number
2.	PropertyClass	X	Char		X	Identifier of the property class
3.	PropertyName	X	Char		X	Identifier of the property
4.	PropertyValue	X	Char		X	Property value

Remarks:

- In this table, article-specific information on fixed or allowed values of selected properties can be specified by assigning one or several values (subsequent table records) to a property of the article.
- For properties, for which there are values stored in the property values table (see section 2.13), this means a restriction of the value set with respect to the discrete values specified in the property value table[22]. This implies that in field 4 only values must be specified, which are also specified in the value table for the given property.
- The value assignments for a property in the article base table prevail over possible proposal values for the property in the property value table (see section 2.13)!

2.13 The property value table

Table name: `PropertyValue`
Obligatory table: yes[23]

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	PropertyClass	X	Char		X	Designator of the property class
2.	PropertyName	X	Char		X	Designator of the property
3.	Position	X	Num		X	Position of the property value
4.	TextID		Char			Text number
5.	RelObjID		Num		X	Relational object number
6.	IsDefault		Bool	1	X	Proposal value?
7.	SuppressTxt		Bool	1	X	Text suppression feature
8.	OpFrom		Char	2		Operator from
9.	ValueFrom		Char			Property value from
10.	OpTo		Char	2		Operator to
11.	ValueTo		Char			Property value to
12.	Raster		Char			Raster (increment)
13.	DateFrom		Date	8		Valid from
14.	DateTo		Date	8		Valid to

Remarks:

- This table lists all possible values for each property.
- The values (fields 9 and 11) are string presentations of numeric values or symbolic (language independent) identifiers. Descriptive (language dependent) names (for a use in the user interfaces) are stored in the table `PropValueText` (see section 2.20). For this purpose, a text number is indicated

[22] Interval values specified in the property value table are not affected by the restriction.
[23] The table can be omitted, if the database is not meant to store configuration data, but e.g. only article texts and prices.

in the 4th field as access key. If no text number is specified or no text available in a required language for the text number, the symbolic (language independent) identifier (from this table) is shown for properties of type 'C', for properties of the other types the respective numeric value.

- A specific value can be indicated only once within a property.
- The relational object number serves as key to access the relational object in the table `RelationObj` (see section 2.15), to which the relationship knowledge for the property value is bound. (If no relational object is required, the number `0` has to be indicated.)
- The value marked as the default (field 6: `IsDefault`) is used for the initialization of the property (see property table, section 2.9)[24].
  If none of the values of a property is marked as the default, the first value is used as initial value for mandatory properties, whereas for optional properties the virtual value "not selected" is used. For the last one, the internal identifier `VOID` is used and reserved for properties of the type 'C', i.e., this must not be used for a real value of such properties.
- The feature in field 7 indicates, whether the property is to be displayed in the description of the article (`0` resp. empty) or not (`1`), if the value is currently selected by the user.
- The property value is specified in the fields 8 to 11. It is possible to indicate an interval for the property value. (This can be used in particular for measurement properties.)
  - A fixed property value is marked by the operator 'EQ'. It does not matter, whether the fields for the From value or the fields for the To value are used. The fields for the respective not used value have to be empty.
  - An open input range is marked by the operators 'GT', 'GE', 'LT' or 'LE' for the From or the To value. The fields for the respective not used value have to be empty.
  - A closed input range (interval) is marked by the operators 'GT', 'GE', 'LT' or 'LE' for the From resp. the To value. Both values have to be set.
  - If the interval value is followed by further individual fixed values for the property, these values are included into the choice list of the OFML property generated for the OCD property. This can be used for standard and proposal values within the interval.
  - If one of the individual values (following the interval value) is marked as default value (field 6), the previous default value is overwritten. Thus, the upper limit could be set as default value for a closed interval.
  - Several interval values can be specified. With the aid of preconditions (see section 2.15), it is then possible to control which intervals in a particular configuration are actually displayed.
- In field 12, an increment can be specified for numeric properties. This increment has to be observed when entering values within the input range determined by fields 8-11.
- Numeric values have to be indicated according to the number of digits resp. decimal digits specified for the property in the property table (see section 2.9). Starting zeros and zeros at the end of the decimal part do not have to be indicated. Examples:
  - Format: 4 digits, thereof 0 decimal digit; value: 1200 → '1200'
  - Format: 3 digits, thereof 1 decimal digit; value: 1.5 → '1.5'
  - Format: 4 digits, thereof 2 decimal digits; value: 20.7 → '20.70' or '20.7'
- If several price lists are displayed in the OCD database but the value or interval is valid only in one price list, the validity period has to be specified in fields 13 and 14.[25].
  If the fields are empty, the value is valid for an indefinite period.

[24] Note: The default value should not be provided with a precondition that is not valid in the initial configuration of the article. Otherwise, the property finally gets another or no value in the initial configuration.

[25] This should match the validity period of the price components for the corresponding price list in the price table (see section 3).

2.14 The property value identification table

Table name: `PropValueIdentification`
Obligatory table: no

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	PropertyClass	X	Char		X	Identifier of the property class
2.	PropertyName	X	Char		X	Identifier of the property
3.	PropertyValue	X	Char		X	Property value
4.	IdentKey		Char		X	Key for identification table

Remarks:

- The table serves to indicate additional identification numbers (of different types) for property values.
- The additional identification numbers are not indicated directly in this table, but indirectly in the table `Identification` (section 2.22) via the key indicated in field 4.
- Additional identification numbers can only be indicated for values, which are specified in the property value table (see section 2.13) and which are no interval values[26].

2.15 The relational object table

Table name: `RelationObj`
Obligatory table: yes[27]

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	RelObjID	X	Num		X	Relational object number (bigger `0`)
2.	Position	X	Num		X	Position of the relation
3.	RelName		Char		X	Relation name
4.	Type		Char	1	X	Type of the relation: 1 - Precondition 2 - Selection condition 3 - Action 4 - Constraint 5 - Reaction 6 - Post-Reaction
5.	Domain		Char	4	X	Scope: C - Configuration P - Price determination BOI - Bill of items PCKG - Packaging data TAX - Taxation schemes

[26] Values entered freely by the user — for properties, which allow this — have to be directly indicated in a respective export, e.g. for the order data exchange.
[27] The table can be omitted, if no relational knowledge is needed.

Remarks:

- This table binds relations to relational objects.
- Presently, the following relation types are possible:
  - Generally, by means of *preconditions* can be determined the validity of the data entities to which the preconditions are bound. Preconditions are possible for the following data entities:
    * The precondition of a *property* determines whether the property is visible for the user (Scope RV) resp. may be evaluated by the user (Scope C).
    * The precondition of a property class determines whether the properties of the class are visible to the user (scope RV) resp. may be evaluated by the user (scope C).
      Preconditions of property classes have priority over any existing preconditions of the properties of the class, i.e., the preconditions of the properties of the class are not even evaluated if the class currently is not valid due to its preconditions.
    * The precondition of a *property value* determines, whether this value is allowed to be set.
      Preconditions of values of configurable properties are evaluated, if the property itself is valid.
      Preconditions of values of non-configurable properties are evaluated if no value is assigned to the property in the article base table (see section 2.12).
    * The precondition of a *component of a bill of items* determines, whether it is allowed to be used.

    If several preconditions are indicated for one entity, the entity is only valid, if all conditions are fulfilled. If no preconditions are specified for a given entity, this entity is generally valid.
  - *Selection conditions* determine, if a property has to be evaluated.
    Selection conditions are evaluated in the consistency check during order list generation for optional properties not currently evaluated as well as for properties for free string input with a currently assigned empty string. If one of the evaluated selection conditions is met, a corresponding error message appears and the order list generation is aborted.
    If several selection conditions are indicated for a property, the property has to be evaluated, if at least one of the selection conditions is fulfilled.
  - *Actions* serve to determine and assign property values or to issue messages to the user.
    Actions on articles and property classes are executed in each configuration step. Actions on properties are executed, if the properties are not hidden due to a precondition[28]. Actions on property values are executed, if the value is set in the current configuration of the article.
  - *Constraints* are used to control and assure the consistency of the configuration of articles. Constraints can also be used to derived values or to restrict value sets. Constraints have to be bound to articles and are executed in each configuration step.
    Not every language, that can be used to code relationship knowledge, supports constraints (see also sections 2.16 and 2.23).
  - Like actions, *reactions* are used to determine and assign property values or to issue messages to the user. But they are not executed in every configuration step, but only in the case of certain events. Reactions can only be associated with articles or configurable properties.
    Reactions of articles are executed once at the start of article initialisation before evaluation of all other relationships.
    Reactions of properties are executed when the value of the property has been changed by the user. The application of the reaction is performed once *before* the evaluation of all other relationships.
  - *Post-Reactions* of articles or configurable properties serve the same purpose as *reactions*, but in contrast to these are executed (once) *after* the evaluation of all other relationships.
    Since no further relations are evaluated after post-reactions, there no changes may be made to the configuration of the article that have an impact on dependent properties! Otherwise the configuration of the article may be left in an inconsistent state.

[28] i.e., this generally applies to properties of scopes R and RG

- The number in field 2 determines the position of the relation in the evaluation sequence. The relations of a relational object with same type (field 4) and same scope (field 5) are evaluated in an ascending order of position numbers, where the number sequence does not have to be continuous. Relations with the same position number are evaluated in an undefined order.
- The scope (field 5) indicates, in what context the relation is to be used:

  C These relations are evaluated when defining the configuration possibilites of a configurable article, both during its initial generation and in each configuration step.

  P These relations are evaluated during price determination (see section 3).

  BOI Relations of this scope control the visibiliy (existence) of a component of a bill of items (see section 2.7).

  PCKG These relations are evaluated when determining packaging data (see section 6).

  TAX These relations are evaluated when determining the taxation information on the basis of taxation schemes (see section 2.25). They are required if the assignment of an article to a tax category is dependent on a particular configuration of the article[29].

  Relations in domains `P`, `PCKG` and `TAX` can only be of the type *action*.

  Additional remark regarding relations in domains `P`, `PCKG` and `TAX`:
  These relations may not affect the current configuration resp. may not have an impacht on subsequent evaluations of configuration relations. Therefore, in these relations assignments are permitted only to internal (auxiliary) properties[30].

The table below gives an overview for which data entities and in which domains the individual relationship types can be used:

Rel.type	Data entity					Domain				
	Articel	P.class	Property	P.value	BOI part	C	P	PCKG	TAX	BOI
Precondition		X	X	X	X	X				X
Selection cond.			X			X				
Action	X	X	X	X		X	X	X	X	
Reaction	X		X			X				
Post-Reaction	X		X			X				
Constraint	X					X				

2.16 The relationship knowledge table

Table name: `Relation`
Obligatory table: yes[31]

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	RelationName	X	Char		X	Relation name
2.	BlockNr	X	Num		X	Code block number
3.	CodeBlock		Char		X	Code block

Remarks:

- The ”knowledge” (the logic) about relationships is stored in this table. Different languages can be used, whose syntax and semantics are described in the appendix. The used language has to be indicated in the version information table (section 2.23).

[29] An example would be the change of the material category in tax type `ECO_FR`.
[30] see also remark regarding persistence and initialization of properties in section 2.9
[31] This table can be omitted, if no relational knowledge is needed.

- Before the evaluation of the relation, the code blocks belonging to a relation are put together to a whole code block according to their number.

2.17 The price table

Table name: `Price`
Obligatory table: yes[32]

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	ArticleID	X	Char		X	(Base) article number
2.	Variantcondition	X	Char			Variant condition
3.	Type	X	Char	1	(X)	Type of price: S - Sales price P - Purchase price
4.	Level	X	Char	1	X	Price level: B - Base price X - eXtra charge price D - Discount
5.	Rule		Char		(X)	Calculation rule
6.	TextID		Char			Text number
7.	PriceValue		Num		X	Price/Amount
8.	FixValue		Bool	1	X	Fixed amount (vs. percentage) ?
9.	Currency	X	Char	3	(X)	Currency of the fixed amount
10.	DateFrom	X	Date	8	X	Valid from
11.	DateTo		Date	8	X	Valid to
12.	ScaleQuantity	X	Num		X	Scale price quantity
13.	RoundingID		Char			Identifier for Rounding Rule

Remarks:

- The base prices and extra charges as well as possible discounts for each article are registered in this table.
- All prices are stated as net prices, i.e. without taxes! The procedure of the price determination is described more fully in section 3.
- If a *variant condition* is indicated for a price item (field 2), this price item is only taken into account in the price determination, if the indicated variant condition is valid. The variant conditions being valid for a certain configuration are determined via price relations (from the tables `RelationObj` and `Relation`).

  Variant conditions have to be written completely in upper case.
- For each price item, both a sales price and a purchase price can be specified (field 3).
- With items for extra charges and discounts (price levels 'X' and 'D', field 4), the joker article '*' (field 1) can be used to specify article-independent extra charges or discounts. In doing so, field 2 (variant condition) must not be empty.
  However, this article-independent table entry is taken into account only if there is no own, specific entry with the same variant condition for the processed article.
- The calculation rule (field 5) modifies the way the price item is used when determining the price:
  - No special calculation rules are currently supported for base prices and extra charges. The amount indicated in field 7 is always added to the price already accumulated during the price determination. Extra charges can be indicated as percentage values. In this case, the absolute amount results from the respective percentage of the base price.

[32] The table can be omitted, if no prices are to be stored.

  - For discounts expressed as a percentage a calculation rule must be specified:
    Calculation rule '1' defines that the price is calculated in relation to the base price.
    Calculation rule '2' indicates that the price is calculated in relation to the price already accumulated during the price determination.

- Via the text number (field 6), an explanation concerning the price item can be indicated in the price text table, e.g. reason for extra charge price. If there is no description for a price item, the application system generates an automatic description, if necessary.
- Field 8 specifies, whether the amount in field 7 is a fixed amount in the currency according to field 9 (`1`) or a percentage (`0`).
- With extra charges (price level 'X'), the amount in field 7 can also be negative. This can be used to represent reduced prices.
- Currencies (field 9) are to be indicated according to ISO 4217, e.g. EUR, CHF, GPB, USD.
- Field 12 serves to indicate scale prices, which become valid starting from a certain number of ordered articles. For this purpose, in field 12 the minimum quantity of ordered articles has to be indicated, which allows to use the table entry. By default, `1` has to be indicated here (no scale price).
  Attention: currently, scale prices are always calculated per order position, but not throughout the whole order.
- In field 13, a rounding rule in table `Rounding` can be referenced (see next section). This rule is then used instead of the standard rounding described in section 3 before the absolute amount of the price item is added to the (current) total price.

2.18 The rounding rule table

Table name: `Rounding`
Obligatory table: no

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	ID	X	Char		X	Identifier of the rounding rule
2.	Number	X	Num		X	Sequential number
3.	Minimum		Char			Minimum amount to be rounded according to this method
4.	Maximum		Char			Amount, starting from which no rounding has to be performed according to this method
5.	Type		Char		X	Rounding method (see below)
6.	Precision		Num		X	Precision of rounding
7.	AddBefore		Num		X	Amount that should be added before rounding
8.	AddAfter		Num		X	Amount that should be added after rounding

Remarks:

- Multiple entries can be created under a rounding rule ID. These are processed in the sequence determined by the number in field 2.
- Fields 3 (Minimum) and 4 (Maximum) determine the range of amounts for which rounding should be performed according to the method specified in field 5, i.e. amounts outside the range determined by fields 3 and 4 are not rounded. Note, that maximum amount specified in field 4 does not belong to the range[33].

[33] i.e. the comparative operator `LT` (*less than*) is used

The fields are declared as optional string fields; they can thus also be empty. If not empty, the fields contain string representations of numeric values[34].
An empty field 3 means a range open downwards whereas an empty field 4 means a range open upwards.

- Because multiple entries can be created under one ID, it is possible to represent several amount ranges. If multiple overlapping amount ranges are specified for the same rounding rule ID, multiple roundings may occur (if the amounts to be rounded are in the overlapping range). Then the processing sequence determined by field 2 is critical, because the result of a rounding is the incoming amount for each subsequent rounding.
- The allowed rounding methods are:

  DOWN Round down

  UP Round up

  COM Commercial rounding (X.5 is rounded up)

  ECOM Expanded (unbiased) commercial rounding (X.5 is rounded up or down to even numbers)
- The precision in field 6 determines the amount by which the rounded amount must be divisible without a remainder. For example, `0.01` rounds to the second digit after the decimal point (and corresponds to the precision of the standard rounding).
- The amounts in fields 7 and 8 can be negative.

Example:

Assuming that a price item should, depending on the nominal amount, be rounded as follows:

- Commercially round amount < 10 EUR to 0.1 EUR
- Commercially round amount < 10 EUR - < 100 EUR to 0.5 EUR
- Round up amount >= 100 EUR to 1.0 EUR and subtract 1 cent.

The table would then have to contain the following entries for the rounding rule:

ID	Nr.	Minimum	Maximum	Type	Precision	AddBefore	AddAfter
R1	1	0.0	10.0	COM	0.1	0.0	0.0
R1	2	10.0	100.0	COM	0.5	0.0	0.0
R1	3	100.0		UP	1.0	0.0	-0.01

[34] equivalent to the values of numeric properties in table `PropertyValue`

2.19 The series table

Table name: `Series`
Obligatory table: no

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	SeriesID	X	Char		X	Series identifier
2.	TextID		Char		X	Text number
3.	CatalogFormat		Char			Format of the catalog data
4.	CatalogDir		Char			Directory of the catalog data

Remarks:

- This table supports applications which do not use the Data Structure and Registration format *DSR* of the company EasternGraphics in the registration of commercial series.
- The text number in field 2 serves as key for the description table `SeriesText` (see section 2.20). The language specific text for a series contains the series designation in the first line. Optionally, further lines with additional explanations can follow.
- In field 4, a directory can be indicated, which contains that data that can be used to choose articles via the user interface. The used data format has to be specified in field 3. The format indication is made up out of the short designation of the format (see below), followed by a hyphen ('`-`') and the indication of the version, composed of main number, point ('`.`') and sub number.
- The following formats can be indicated in field 3:

Short description	Explanation
OAS	OFML Article Selection (OFML Part V)
XCF	eXtensible Catalog Format (Company EasternGraphics)

2.20 The description tables

All text tables

Article short descriptions: `ArtShortText` (Obligatory table)
Article long descriptions: `ArtLongText`
Property class names: `PropClassText`
Property names: `PropertyText` (Obligatory table)
Hints on properties: `PropHintText`
Property group names: `PropGroupText`
Property value names: `PropValueText`
Price texts (Explanations concerning price items): `PriceText`
Additional texts for BOM positions: `BillOfItemsText`
Messages for the user: `UserMessage`
Series names: `SeriesText`
Descriptions for classifications: `ClassificationText`

have the same structure:

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	TextID	X	Char		X	Text number
2.	Language	X	Char	2	X	Language
3.	LineNr	X	Num		X	Line number
4.	LineFormat		Char		X	Line format control
5.	Textline		Char		X	Text line

Remarks:

- The access is effected with the help of text numbers assigned in the respective tables.
- The language has to be indicated according to ISO 639-1, e.g. 'de' (German), 'en' (English), 'fr' (French)[35].
- A text consists of one or multiple lines. However, multiple lines are allowed only for tables `ArtLongText`, `PropHintText`, `UserMessage` and `PropValueText`. (For other text types, additional lines are not processed.)
  With table `PropValueText` the lines are processed in combination with field *TxtControl* of table `Property` (section 2.9).
- In the composition of texts, the recommendations of the OFML Standards Committee should be taken into account for uniform composition of customer-oriented article descriptions.
- Field 4 contains formatting hints for form layout modules of an OFML-based distribution system. The appearance of the text can thus be controlled in some respects.

  The following formatting codes are possible:

  \ (Line feed)

  The text line is output in a new line. This is the default.
  With a property text control code `0`, this code is ignored in the first line of the property value text and instead the code ~ (see below) is used.

  ~ (Continuous text)

  Die Textzeile wird als Flie"stext an den vorherigen Text geh"angt. Beginnt die Textzeile selber nicht mit einem Leerzeichen, wird dieses vom Formularlayout–Modul eingef"ugt.

  The line is attached to the previous text as continuous text. If the line itself does not begin with a space, this is inserted by the form layout module.

  ^ (Conditioned continuous text)

  Used in text segments comprising various text types, e.g. property name plus value description with property text control code `0`: If the line (including the separator to be inserted as required) still fits in the composite text segment without exceeding the line width required by the form layout module, it is attached to the previous text as continuous text. Otherwise a line feed is performed at the start of the line.

2.21 Value combination tables

Some languages to code relationship knowledge (see section 2.16) allow the use of value combination tables. In relationship knowledge, value combination tables are used to check the consistency of a value combination, to derive values or to restrict the value set of a (restrictable) property.

A value combination table specifies all possible value combinations with respect to a defined set of properties.

Ex.:

	DESIGN_GROUP	COLOUR_CORPUS
1	A	F001
2	A	F002
3	B	F002
4	B	F003

The file name of a value combination table is made up out of the identifier, under which the table is called in relationship knowledge, the addition "`_tbl`" and the suffix "`.csv`". In doing so, the table name is entirely written in small letters.

[35] For the sake of consistency throughut all OFML data, the 2-digit language code was chosen. When exporting the data into a format which uses a 3-digit codification according to ISO 639-2, e.g. BMEcat, the respective application is responsible to perform a conversion.

Ex.:

Table identifier: `COLOURS_CORPUS`
Dateiname: `colours_corpus_tbl.csv`

The table definition is the same for all OCD value combination tables:

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	LineNr	X	Num		X	Number of the table line
2.	PropertyName	X	Char		X	Name of the property
3.	Value	X	Char		X	Property value

The fields of a line of the logical value combination table (see example above) are combined via their (imaginary) line number[36].

Both property names and property values have to be written completely in upper case.

Ex.:
The (logical) value combination table from the example above is represented in an OCD value combination table as follows:

```
1;DESIGN_GROUP;A
1;COLOUR_CORPUS;F001
2;DESIGN_GROUP;A
2;COLOUR_CORPUS;F002
3;DESIGN_GROUP;B
3;COLOUR_CORPUS;F002
4;DESIGN_GROUP;B
4;COLOUR_CORPUS;F003
```

If the value combination table includes an attribute which refers to a restrictable property (see section 2.9), it is possible to specify a value set in the field of a table line for this property.

Ex.:
Assuming that the property `COLOURS_CORPUS` from the example above is restrictable, the logical value combination table and the corresponding OCD value combination table could look like this:

	DESIGN_GROUP	COLOUR_CORPUS
1	A	F001, F002
2	B	F002, F003

```
1;DESIGN_GROUP;A
1;COLOUR_CORPUS;F001
1;COLOUR_CORPUS;F002
2;DESIGN_GROUP;B
2;COLOUR_CORPUS;F002
2;COLOUR_CORPUS;F003
```

[36] This means, it is not necessary to provide an own table definition available each (logical) value combination table.

2.22 The identification table

Table name: `Identification`
Obligatory table: no

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	EntityID	X	Char		X	ID of the entity (article, property, value)
2.	Type	X	Char		X	Type of the identification number
3.	IdentNr		Char		X	Identification number

Remarks:

- The table serves to specify additional identification numbers for articles, properties and property values.
- The manner resp. the context of the use of an identification number is determined by its type (field 2). Currently, the following types are allowed (see also terms in the appendix)[37]:

Type	Explanation
CustomID	Dealer resp. major client-specific article number
EAN.UCC-8	8-digit ID according to EAN.UCC
EAN.UCC-13	3-digit ID according to EAN.UCC
EAN.UCC-14	14-digit ID according to EAN.UCC
GLN	Global location number
Intrastat	Intrastat number
CustomsTarif	Customs tariff number

The type *CustomID* is used, if the data base is designed for a specialist dealer or major client, who uses an article number deviating from the manufacturer's one. If applicalbe, the OFML application then has to indicate/use the customer-specific article number.

2.23 The version information table

Table name: `Version`
Obligatory table: yes

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	FormatVersion		Char		X	Number of the used OCD format version
2.	RelCoding		Char		X	language used for relationship knowledge
3.	DataVersion		Char		X	Database version
4.	DateFrom		Date	8	X	Usable from
5.	DateTo		Date	8	X	Usable to
6.	Region		Char		X	Sales region
7.	VarCondVar		Char			Variable for variant conditions
8.	PlaceHolderOn		Bool		X	Placeholder in IN–comparisons?
9.	Tables		Char		X	included tables
10.	Comment		Char			free comments, additional information

[37] Not all of the ID types are applicable for each entity type.

Remarks:

- The table serves to give information concerning the used format and the product database. A version control system or other systems can thus determine the structure and the usability of the database.
  The table may contain only one entry.
- The OCD format version (field 1) has to be indicated in the form `MajorNumber.MinorNumber` according to the OCD format specification.
- The language used to code relationship knowledge has to be specified in field 2 (see section 2.16). The following languages can be used: `OCD_1`, `OCD_2`, `OCD_3`, `OCD_4`, `SAP_LOVC`. The description of the languages can be found in the appendix.
- The database version (field 3) has to be indicated in the form `MajorNumber.MinorNumber.BuildNumber`. The manufacturer can freely assign the numbers, but has to assign them in a strictly monotonously ascending order.
- The identifier for the sales region[38] (field 6) can be freely assigned. However, it has to correspond to the identifier, with which further required data (geometry, catalog) concerning the sales region is referenced in the respective software system.
- In field 7, the variable is specified, that is used in price relationships instead of `$VARCOND` (OCD Language Sets) resp. `$self.variant_condition` (SAP Language Sets) to assign variant conditions (see also section 3).
  For the identifier of the variable all alphanumeric characters including the underscore are allowed. The first character must not be numeric.
  When using the variable within relationships in the table `Relation`, the identifier has to be preceded with the dollar sign ('`$`').
- Field 8 specifies whether placeholder characters in string constants in IN–comparisons should be replaced.
- The tables, currently contained in the database, are listed in field 9, separated by a comma. This also concerns the obligatory tables, but not the value combination tables. The tables names have to be indicated according to the specification of the used OCD format version (field 1)[39].
  Additional blank characters after the commas are allowed.

[38] major clients with specific price lists or deviating configuration data are also represented by the *sales region* concept within this context.

[39] that means without the prefix `ocd_` and without the suffix `.csv`.

2.24 The code scheme table

Table name: `CodeScheme`
Obligatory table: no

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	SchemeID	X	Char		X	Unique identifier to reference the scheme
2.	Scheme		Char			Description of the scheme
3.	VarCodeSep		Char			Character string to separate base article number and variant code (only for pre-defined schemes)
4.	ValueSep		Char			Character string to separate property values (only for pre-defined schemes)
5.	Visibility		Char	1		Visibility mode – indicates, which properties are meant to be included in the variant code (only for pre-defined schemes): `0` – only the currently valid and visible properties `1` – all configurable properties
6.	InVisibleChar		Char	1		Replacement character for currently invalid resp. invisible properties: As many characters as indicated for the property in the length field of the property table are displayed. If the field is empty, '`-`' is used.
7.	UnselectChar		Char	1		Replacement character for currently not evaluated/selected optional or restrictable properties: As many characters as indicated for the property in the length field of the property table are displayed. If the field is empty, '`X`' is used.
8.	Trim		Bool	1	X	Trim character – indicates, whether the individual property values are meant to be displayed exactly according to the indication in the length field of the proerty table (`0`), or whether not assigned digits (blank characters) at the end can be deleted (`1`).
9.	MO_Sep		Char			Character string to separate the values of mulitvalued properties
10.	MO_Bracket		Char	2*N		Characters to be used as brackets for multivalued properties

Remarks:

- The table serves to indicate codification schemes and parameters for the generation of final article numbers.

- Final article numbers are conceptually composed of the base article number and the so-called *variant code*. The concrete position of base article number and variant code in the final article number is determined by the individual codification schemes. In the variant code, the current values of the configurable properties are coded.
- The codification scheme, which is to be used for an article, is determined in the article table (section 2.2) with the help of the scheme identifier[40].
- A difference is made between *pre-defined* and *user-defined* codification schemes. The respective codification procedures are fully described in the section 4.
- The fields 3 to 10 serve to parameterize the codification schemes as described in the section 4.
- The string in field 3 must not be empty (for pre-defined schemas). If the field is still empty, a string consisting of a single space is used.
- In order for a given article setup to be definitly reconstructed based on its final article number, in the fields 6 and 7 such replacement characters have to be used that would *not* result in a regular value of optional or restrictable properties.
- If the Trim flag is set in field 8, the contents in fields 6 and 7 has to differ from the contents in field 4, and field 4 must not be empty.
- The Trim flag must not be set if the article has properties for which the user can freely enter values. Otherwise correct processing of final article numbers cannot be guaranteed in all cases.

2.25 Taxation Schemes

For articles, no concrete tax rates are stored in the OCD data, but each article is assigned to a so-called taxation scheme, which, for each country[41] and for every relevant tax type, describes the respective (abstract) tax category to be used. The current legal tax rates for each tax category are then managed in the OFML applications themselves and used for form creation.

Table name: `ArticleTaxes`
Obligatory table: no

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	ArticleID	X	Char		X	(Base) article number
2.	TaxID		Char		X	ID of the taxation scheme
3.	DateFrom	X	Date	8		Valid from (incl.)
4.	DateTo		Date	8		Valid to (incl.)

Remarks:

- This table assigns an article to a taxation scheme.
- If no assignment to a taxation scheme is made for an article, (e.g. if the tables described in this section don't exist at all), a taxation scheme established or set in the application system is used.
- In fields 3 and 4, due dates can be represented for which legal changes regarding the assignment of articles to tax categories become effective. If there are multiple entries for an article in the table, these two fields must not be empty (and have to specify non-overlapping periods).

[40] Principally, an individual codification scheme can thus be defined for each article.
[41] of the distribution area represented in the OCD database

Table name: `TaxScheme`
Obligatory table: no

No.	Name	Key	Type	Length	Oblig.	Explanation
1.	TaxID	X	Char		X	ID of the taxation scheme
2.	Country	X	Char		X	Country code (ISO 3166-1)
3.	Region	X	Char			Region code (ISO 3166-2)
4.	Number	X	Num		X	Sequential number
5.	TaxType		Char		X	Identifier of tax type
6.	TaxCategory		Char		X	Identifier of tax category

Remarks:

- This table defines, country–specifically and, where necessary, also region–specifically, the taxation schemes relevant for the articles of the OCD database.
- Field 3 (*Region*) can be used to specify specific regulations for regions within a country. The entries for a given taxation scheme and a given country in which field 3 is empty describe the overall regional normal regulations for the country. The complete code as per ISO 3166-2 should not be given as the code in this field, but only the region-specific part after the country code and separator (e.g. "'TH"' for Thuringia instead of "'DE-TH"').
- The sequential number in field 4 is relevant for cases in which multiple tax types are raised for the article. The number then specifies the sequence in which the respective tax rates are applied. In the application sequence however, the particular OFML application has the final power of decision, because it must, as necessary, give legal regulations priority.
- The tax type and tax category identifiers (fields 5 and 6) can be freely assigned in principle. However, this is only useful for specific OFML applications. In order to guarantee consistent use in various OFML applications, the identifiers for the usual tax types and tax categories are standardised. This is done in a separate appendix to the OCD specification (see app. H), which, if required, can be dynamically expanded by request and after approval by the OFML Standardisation Committee.
- If an article is not assigned to a taxation scheme or if the scheme contains data for neither a requested region nor a requested country, the standard category (with the applicable tax rate in the country) is used by the OFML application in the tax type *VAT*.
- Using the function `SET_TAX_CATEGORY` (see D), in `TAX` relations an article can be assigned to a tax category for a given tax type different from the taxation scheme, depending on certain variants of the article.

3 The price determination

This section describes how the price of an article in a concrete configuration is determined with the help of the price table and relationship knowledge.

3.1 Overview

Conceptually, the total price (final price) of an article for a given price type (sales vs. purchase) consists of *price items* at different levels. Price items are divided into unconditioned and conditioned ones. *Unconditioned* price items apply regardless of the configuration of the article. *Conditioned* price items, in contrast, only apply for certain configurations of the article. The condition is formulated using a *variant condition* (field 2 in price table), which has to be assigned accordingly in price relationships (see section 3.2).

The method described below in the same way applies to both price types.

For a given price item there may be multiple entries at a given price level, differing in their periods of validity, currencies and/or other attributes. The most appropriate of these table entries is determined depending on the parameters of current price calculation (see section 3.3).

The relevant price items at the different price levels (field 4) are determined according to the following order:

1. Base prices (level 'B')
2. Extra charges (level 'X')
3. Discounts (level 'D')

With each determined (valid) price item the total price for the article will be accumulated, considering the calculation rule (field 5 in the price table). In doing so, if necessary, the determined absolute amount will be rounded to 2 positions after the decimal point according to the method of commercial rounding (X.5 is rounded up), provided that no separate rounding rule is stated for the price item (field 13).

At the end, the taxes are determined for the article. In doing so, data from the associated taxation schemes is used (see section 2.25). The exact manner of the tax amount determination (order-wide or per position) and the manner of showing it in the form depend on the respective OFML application.

3.2 Relevant table entries

Within a price level, the relevant table entries are determined as follows:

1. Determination of the price items for the article without variant condition.
2. Evaluation of all relevant price relationships and determination of the price components for the article with the variant conditions, which are applied in these price relations. Price relationships are marked by the scope 'P' in the table `RelationObj` (field 4) and have to be of the type action ('3') (field 3). Variant conditions are applied in price relationships (table `Relation`) by assigning the name of the variant condition to the special variable `$VARCOND` (OCD language sets) resp. to the auxiliary property `$self.variant_condition` (SAP language sets)[42]. Both `$VARCOND` and `$self.variant_condition` are treated like multivalued properties, i.e., in a single price relationship these variables can be assigned multiple names of variant conditions.

   Before accessing the price table, the variant conditions applied in the price relationships are converted to upper case[43].

   The relevant price relationships are determined according to the order mentioned below from the relational objects

[42] Instead of these standard variables, a separate specific variable can also be used that should then be specified in the version information table (see section 2.23).

[43] see also note on variant conditions in the price table, section 2.17

1. of the article
2. of the property classes of the article
3. of the currently assigned properties of the article[44]
4. of the value of currently assigned properties.

If a variant condition is assigned in various price relations, it is not defined[45], whether the price item associated with the variant condition will be applied for each of these relations or only once[46].

Along with a variant condition being assigned, price relationships can also include evaluation of auxiliary properties for subsequently evaluated price relationships. However, such an assignment is only effective during evaluation of price relationships within a price level.

3.3 Valid table entries

From the table entries read-out off the price table for a given price item (see section 3.2) the valid (most appropriate) entry is determined as follows:

1. Entries for level 'B' (base price) are ignored if they do not contain a fixed amount (but a percentage).
2. Entries without a correct date indication in the fields 10 and 11 (period of validity) or with a period of validity which is not fulfilled for the required price calculation date, are ignored.
3. If the price is supposed to be determined in a certain currency, only the entries with that currency are taken into account. If none of the (timely valid) entries has the required currency, all entries are included in the the further choice.
4. Those entries whose quantity indication for a scale price (field 12) is smaller than the quantity of the current order or contract position, are excluded from the remaining entries.
5. From the remaining entries the entry with the latest date in field 10 ("Valid from") is used.
   (If there are multiple entries even with respect to this criterion, it is not defined which of them will be used by the OFML application[47].).
6. If the calculation rule or the fixed price/percentage indication is not allowed for the specified price level in the remaining entry, this entry will be ignored, i.e. the desired price item cannot be determined.

3.4 Price factors

In the relationship knowledge (table `Relation`), price factors can be indicated for a price item, which is linked with a variant condition. For that purpose, the function *$SET_PRICING_FACTOR()* is used, which is specified in all OCD language definitions as follows:

- *$SET_PRICING_FACTOR(<Variant condition>, <Factor>)*

  The function defines the <Factor>, with which the price item has to be multiplied, associated with the stated <variant condition> in table `Price`, if that <variant condition> currently is assigned to variable `$VARCOND`.

  <Variant condition> can be specified as an expression whose evaluation yields a character string.

  <Factor> is an arithmetical expression. (If the result of the expression is undefined, the function has no effect.) Price factors can have a negative amount.

  The call of the function usually follows the assignment of the variant condition to the variable `$VARCOND` within the same relationship.

[44] including non-configurable properties
[45] i.e., depends on the OFML application
[46] Therefore, creating such relationships is not recommended.
[47] If – considering correct data – there are no 2 entries for a price item and a given currency with the same date in field 10, this case only can occur if there are entries for the price item with different currencies (and the same date in field 10), but none of these currencies matches the requested currency. Then it is not apparent for the application which is the most appropriate table entry.

Example:

If the value "Set 1" is assigned to the property "Electrification", the extra charge price is supposed to be determined depending on the width of the table. The price relation, which is bound to the relational object for the value "Set 1" of the property "Electrification", could then be defined as follows:

```
$VARCOND = 'ABC123_ELECTR_1', $SET_PRICING_FACTOR('ABC123_ELECTR_1', WIDTH / 1000)
```

Remark:
Since the property WIDTH in indicated in mm, the extra charge price (from the price table) is multiplied with the current width in meters.

The function call can be linked to a condition. The factor then will be applied only, if the condition is definitely fulfilled.

Example:

If the value "Set 2" is assigned to the property "Electrification", the extra charge price is supposed to be increased by 10 percent, if the width of the table is more than one meter. The price relation, which is bound to the relational object for the value "Set 2", could then be defined as follows:

```
$VARCOND = 'ABC123_ELECTR_2',
$SET_PRICING_FACTOR('ABC123_ELECTR_2', 1.1) IF WIDTH > 1000
```

In the SAP language sets, the function *$SET_PRICING_FACTOR()* has two additional parameters at the beginning, to which only a fixed value can be assigned in OCD:

- *$SET_PRICING_FACTOR($self, variant_condition, <Variant condition>, <Factor>)*

If multiple calls are made in a price relationship for setting a factor for one and the same variant condition, only the last call is effective.

4 The final article number generation

The final article number for an article is generated according to the scheme which is indicated for the article in the article table by the scheme identifier. If no identifier or an identifier is indicated, which is not referenced in the scheme table (see section 2.24), no special final article number is generated for the article. The final article number is then identical to the base article number.

In this section the following simple example is used for explanations:

> A cupboard with the base article number `0815` belongs to the property class `Cupboard` and has currently the following properties:
>
> ```
> Surface: 03
> Hight: 5H
> Accessories (optional): not selected
> ```
>
> The property `Lock` is currently not visible (invalid). The field length for all properties shall be 2.

4.1 The predefined schemes

In these schemes, the final article number principally starts with the base article number from the article table. Then follows the character string, indicated in the 3rd field of the scheme table. Finally, the separation character string is followed by the variant code, which is generated in the predefined schemes as described below.

The identifier for a predefined scheme (see below) has to be indicated in the 2nd field of the scheme table.

The separation character string "`-`" is assumed for the examples.

- **KeyValueList**

  Every currently valid property is displayed according to the order of the property table as follows:

  `<Property class>.<Property>=<Property value>`

  The semicolon is used as separation character between the properties. For currently not selected optional properties, the internal value identifier "`VOID`" is used.

  The parameter fields 4 to 7 are of no significance for this scheme. `1` is always used as trim character (regardless of the indication in field 8).

  > The example then is as follows:
  >
  > `0815-Cupboard.Surface=03;Cupboard.Hight=5H;Accessory=VOID`

- **ValueList**

  The properties are displayed according to the order of the property table solely by means of the current property value.

  The presentation can be controlled with the help of the parameter fields 4 to 10. Field 5 (visibility mode) should always be filled when new data is created in order to achieve a defined behaviour.

  > For the example, we assume a void character string as character string to separate the property values (field 4) . For the replacement characters (fields 6 and 7), the standard characters '`-`' resp. '`X`' are used. (The trim character is of no significance, because all values have exactly the length indicated in the property table.)
  >
  > In the visibility mode `0`, the example is then :
  >
  > `0815-035HXX`
  >
  > and in the visibility mode `1`:
  >
  > `0815-035HXX-`

4.2 User-defined schemes

Codification rules, deviating from the predefined schemes, can be defined in field 2.

The syntax of the scheme description is:

<Scheme> := [<PropertyClass>:<PropertyName> | <TableCall> | @ | <Char>,] 1:n

<PropertyClass> := Name of the property class in the table Property

<PropertyName> := Name of the property in the table Property

<TableCall> := Table call according to syntax and semantics in language definition `OCD_2`

<Char> := A single character (except '`,`' and '`@`')

To generate the final article number, the scheme description is processed from the left to the right, executing the following replacements:

- *<PropertyClass>:<PropertyName>* is replaced by the current property value. In doing so, the formattings from the property table and the replacement and formatting instructions from the fields 5 to 10 of the scheme table are taken into account. If field 5 (visibility mode) is empty, mode `1` is used.
- In the table call, actual parameters that refer to a property are replaced by the current value of the property. (The special actual parameter `$BAN` is replaced by the base article number.) If the addressed value combination table contains exactly one logical line matching the actual parameters and if this line contains a value for the special table parameter `$FAN`, this value is used at the current position of the final article number. (If the table call delivers no or a non-unique result for `$FAN`, the behaviour of the OFML application is undefined.)
- *'@'* is replaced by the next character of the base article number. The base article number is processed from the left to the right, too.
- *<Char>* is not replaced. The indicated character is inserted at the current position.
  All printable characters from the ASCII character set may be used, except '`,`' and '`@`'.

  In the scheme description, the example then is:

  `Cupboard:Hight,_,@,@,-,@,@,_,Cupboard:Surface`

  the final article number:

  `5H_08-15_03`

Note:

User-defined final article numbers are of limited suitability for the reconstruction of a product configuration. For this purpose, in principle, all configurable properties have to be encoded, observing the restrictions regarding fields 6 to 8 referred to in section 2.24. *Not* suitable for reconstructing a product configuration is a user-defined scheme, if it uses a table call, or if the Trim flag is set and two properties in the scheme follow another directly, i.e. are not separated by characters.

4.3 Multivalued properties

In all codification types, the presentation of the current values of multivalued properties is controlled with the fields 9 and 10 of the scheme table:

- Principally, all currently set values are displayed according to the order which is determined by the position indication in the property value table (see section 2.13).
- The character string indicated in field 9 is used to separate the values. If the field is empty, a comma (",") is used. If exactly one character is indicated, for the predefined scheme `ValueList` it must not be identical to the separation character from field 4 of the scheme table.

- If field 10 is not empty and contains an even number of characters, the first half of the character string is placed in front of the value list and the second half is attached to the value list. A bracket presentation can thus be realized.

  Example: "`[ABS,ZV]`"

5 Property text control

This section describes how the generation of the text, which describes the property in commercial forms (bill of items and others), can be controlled with the help of the control code in the field *TxtControl* in the property table (see section 2.9).

Principally, the text describing the property is made out of the (single line) name of the property (from table `PropertyText`) and out of the text of the currently assigned value. Property value texts (table `PropValueText` or freely entered values with properties of type T) can be multiline. According to the respective control code, not all lines are used for the property description.

In contrast to the property descriptions in commercial forms, the appearance of a property in components of (graphical) user interfaces to evaluate properties (property editors) cannot be influenced. A property (except for type T) there is always displayed by its name and the first line of the text of the currently assigned value.

The following codes can be used[48]:

0 The first line of the property description is made out of the property name and the first line of the text of the current property value[49]. The remaining lines of the property value text follow.

For multivalued properties (see field *MultiOption* in the property table, section 2.9), the respectively first lines of the texts of the currently set values are lined up. The order is determined by the position indication in the property value table (see section 2.13) and the indication from the field *MO_Sep* of the scheme table (section 2.24) is used as separation character string between the values (resp. ”,”, if the field is empty).

This is the standard procedure, in particular for single line property value texts.

1 The property description corresponds to the (multi line) text of the current property value, that means the usual property name in the first line (standard) is suppressed.

Example:
A chair has the property ”Mechanics” with the values:
”Standard mechanics with gas pressure spring” (1 line)
”Synchronous mechanics Optima Plus” (1 line)
If the first value is selected, the property description in the form then reads
”Standard mechanics with gas pressure spring”
instead of
”Mechanics: standard mechanics with gas pressure spring”
in the standard case.

2 The first line of the property description consists of the property name only. The remaining lines correspond to the lines 2..n of the text of the current property value.

Example:
A cupboard has the property ”Shelves strengthened” with the values ”Yes” and ”No”.
The field *SuppressTxt* in the table `PropertyValue` is set on True for the value ”No”. If this value is selected, there appears no text in the form for this property.
For the value ”Yes”, ”SuppressTxt” is set on False. The following property description appears
”Shelves strengthened”
instead of
”Shelves strengthened: Yes”
in the standard case.

3 The property description is made out of the lines 2..n of the text of the current property value, that means property name and first line of the property value text are suppressed.

[48] For properties of type T only codes `0` and `5` are allowed.
[49] The two text blocks are separated by a colon and a space, if these characters are not included already at the end the property name.

Example:

A table has the property ”Electrification” and the following text is given for its value `E01`:

Line 1: Set 1
Line 2: Electrification consisting of:
Line 3: - 2x Cable snake
Line 4: - 2x Multiple socket

The property editor displays: ”Electrification: Set 1”.

However, the property description reads:

”Electrification consisting of:
- 2x Cable snake
- 2x Multiple socket”

4 No description of the property in the form.

The same effect can be achieved, if the field *SuppressTxt* is set on True for all values of the property in the table `PropertyValue`.

It is senseful to use this mode for example for auxiliary properties which can be configured by the user, but are not meant to be printed.

Example:

A chair has the property ”two-colour cover” with the values ”Yes” and ”No”. If ”No” is selected, the property ”Cover colour” is valid. If ”Yes” is selected however, the properties ”Seat colour” and ”Back colour” are valid.

The property ”Two-colour cover” shall not be described in the form in this case. This is done by code 4.

5 Valid for multivalued properties and properties of type T.

The effect for multivalued properties is:
The first line of the property description consists of the property name. The (possibly multi-line) texts of the currently set values follow according to the order which is determined by the position indication in the property value table (see section 2.13). The value texts are separated by the character string indicated in the field *MO_Sep* of the scheme table (section 2.24) (resp. ”,”, if the field is empty).

The effect for properties of type T is:
The first line consists of the property name. The following are the lines of text currently entered by the user

6 The determination of packaging data

This section describes, how to determine the packaging data (dimensions, volumes, weights, number of packaging units) of an article in a concrete configuration with the help of the packaging table (section 2.5) and relationship knowledge (sections 2.15 and 2.16).

The calculation of the data is done in the following steps:

1. Determination of the basic data of the article in the basic configuration by reading out the entry without variant condition (empty field 2) from the table `Packaging`.
2. Determination of deviating data with the help of variant conditions:
    1. Determination of all the variant conditions, which are valid for the current configuration, with the help of the packaging relations.
    2. For each determined currently valid variant condition:
    Fetching the corresponding entry from the table `Packaging` and addition of the amounts of all non–empty fields to the corresponding amounts from the basic data resp. to the amounts possibly accumulated by previous variant conditions.

Regarding step 2.1:

- Packaging relations are marked by the scope 'PCKG' in the table `RelationObj` (field 5) and have to be of the type action ('3') (field 4).
- Variant conditions are applied by assigning the name of variant condition to the special variable `$VARCOND` in the packaging relations (table `Relation`).
- Variant conditions can be linked with a factor. For that purpose, the following function is used[50]:

  $SET_PCKG_FACTOR(<Variant condition>, <Data item>, <Factor>)

  The function defines the <Factor>, with which the data item has to be multiplied, which was fetched from the packaging table for the given <Variant condition>.

  <Variant condition> can be specified as an expression whose evaluation yields a character string.

  <Data item> is the name of the concerned field in the packaging table, enclosed in inverted comma and completely written in upper case, e.g. 'NETWEIGHT'.

  <Factor> is an arithmetical expression. (If the result of the expression is undefined, the function has no effect.) Factors can have a negative amount.

  The call of the function usually follows the assignment of the variant condition to the variable `$VARCOND` within the same relationship.

  The function call can be linked to a condition by means of an `IF`–expression (s. app. B). The factor then will be applied only, if the condition is definitely fulfilled.
- The relevant packaging relations are determined according to the order mentioned below from the relational objects
    1. of the article
    2. of the property classes of the article
    3. of the currently assigned properties of the article[51]
    4. of the value of currently assigned properties.

[50] same all OCD language definitions
[51] including non-configurable properties

A Language definition `OCD_1`

- This simple language allows to indicate conditions (all types of relations) and of assignments to the properties of an article (within actions, reactions and post-reactions). Several assignments can take place within an action, reaction or post-reaction. They have to be separated by a comma.
- Character string constants have to be enclosed in single quotation marks (').
- With keywords as well as with identifiers for properties and values of string properties, upper/lower case is ignored.
  Example: `IF` is identical with `if`.
- In logical and arithmetic expressions, property names can be used as variables. During evaluation of the expression, they are replaced by the current value of the property. The special variable `$BAN` is replaced by the base article number of the current article.
- **Arithmetic expressions**:
  - Arithm. expression may be used
    - as numeric operands in comparisons,
    - on the right side of assignments and
    - as multipliers in the built-in functions *SET_PRICING_FACTOR()* and *SET_PCKG_FACTOR()*.
  - A complex arithmetic expression may be built from subordinated arithmetic expressions with the help of the basic arithmetic operations and parentheses.
  - Basic arithmetic expressions are:
    - numeric constants
    - references to numeric properties
    - calls of arithmetic functions (see appendix F)
  - If a basic arithmetic expression refers to a property that the processed article does not possess or that has no value assigend, the result of the expression is *undefined*. If an undefined arithmetic expression is part of a higher-level complex arithmetic expression, the result of that expression is undefined, too.
- **Conditions**:
  - Conditions are basic or complex Boolean (logical) expressions. A logical expression is evaluated either as *true* or *false*. Sometimes a logical expression cannot be evaluated, the result is then *undefined* (see below).
    Complex Boolean expressions are built with the help of the operators `AND` and `OR` from subexpressions.
    In concatenated `AND` and `OR` operators, the `AND` concatenations are evaluated first. The evaluation order can be controlled by using parentheses. Compare for example `A and B or C` with `A and (B or C)`.
  - Basic logical expressions are:
    - Comparisons
    - Negation
    - Special conditions
  - Comparisons are stated with the help of the known comparison operators: < (or `LT`), <= (or `LE`), = (or `EQ`), <> (or `NE`), =>[52] (or `GE`) and > (or `GT`).
    The operands on both sides of the comparison have to be of the same type.
  - String comparison is based on the lexicographic order, i.e. two strings are compared character by character until two differing characters are found. The string, whose differing character is lexicographically smaller than the corresponding character of the other string, then is considered to be the smaller string. If a string has been completely processed before a difference is

[52] alternatively the form => may be used

detected, the shorter string then is considered to be the smaller string.
The lexicographic order of the characters is determinded by the coding of the characters in the used character set. Therefore, with character set ISO -8859- 1 (Latin -1), used in OCD, for example applies `'A' < 'a'`.
The lexicographic ordering is particularly to be considered regarding strings that consist entirely of digits. There the comparison may produce a result differing from the result produced by the comparison of the corresponding numbers. For example applies `900 < 1000` but `'900' > '1000'` !

- Logical expressions can be negated by means of the `NOT` operator.
- *Special conditions* are:
  * `SPECIFIED <Property name>`
    This condition is true, if the article possesses the indicated property and if a value is assigned to it.
  * `<Property name> IN (<Value set>)`
    This condition is true, if the current value of the property indicated in the left operand is comprised in the value set indicated in the right operand. The values in the value set have to be separated by commas.
- If a basic logical expression refers to a property that the article in question does not possess or that has no value assigend, the expression cannot be evaluated and the result is *undefined*. The only exception is the `SPECIFIED` expression, which can be used to prevent undefined logical expressions.
  The following rules then apply for the operators `AND` and `OR`:
  * The result of an `OR` operation is undefined, if either both subexpressions are undefined, or if one subexpression is undefined and the other is not true. (If at least one subexpression is true, then the `OR` link is true in any case, even if the other subexpression is undefined.)
  * The result of an `AND` operation is undefined, if either both subexpressions are undefined, or if one subexpression is undefined and the other is true. (If at least one subexpression is not true, then the `AND` link is not true in any case, even if the other subexpression is undefined.)

  The following rules apply for the different relation types concerning undefined logical expressions:
  * A *precondition* is violated, if it is definitely false, that means it is not violated, if the logical expression is undefined.
  * A *selection condition* is violated, if it is not definitely true, that means it is also violated, if the logical expression is undefined.

- **Assignments**:
  - Assignments are performed via assignment operator `=`. The left operand is a property or the special variable `$VARCOND` (see section 3).
  - The operands on both sides of the assignment must be the same type (string vs. number).
  - If the left operand is a numeric property, the right operand, if need be, is rounded according to the number of allowable decimal digits defined for the property. In this process, the method of mathematical rounding is applied.
  - If there is an undefined arithmetic expression on the right side of the assignment, the operator has no effect.
  - An assignment can be equipped with a condition. This condition has to be indicated after the keyword `IF`. The assigment will only take place, if the condition is definitely fulfilled (not, if the logical expression is undefined).

B Language definition OCD_2

The language definition includes all determinations from the language definition OCD_1. Additionally, the following further determinations are valid:

- Character strings can be concatenated with the help of the operator +.
  This produces *string expressions*, where character string constants and values of string properties can be linked together.
- Numeric values can be converted into a character string with the help of the function STRING():
  The result string contains the simple decimal point notation of given number with the dot ('.') as decimal separator and no thousands separator or other delimiters. Non-significant fractional parts are not presented, e.g. 9.0 → "9".
  The function can be used in string expressions.
- The Boolean constants FALSE and TRUE can be used in logical expressions.
- With regard to conditions, the following concretisations apply for *restrictable properties*:
  - The condition SPECIFIED is fulfilled, if the value set for the property has been restricted to exactly one value.
  - A comparison (including the IN condition) is only possible, if the value set has been restricted to exactly one value (otherwise the result of the expression is undefined).
- Syntax and semantics are defined for the relation type *Constraint*. They are described more fully below in B.1.
- Checks as well as value assignments in preconditions, constraints, actions, reactions and post-reactions can be realized with the help of value combination tables and the function TABLE().
  The respective syntax and semantics are described more fully below in B.2.

B.1 Constraints

- A constraint is a complex language construct, which is mainly used to control the consistency of a configuration, but can also be used to assign values to properties or to restrict value sets.
- Constraints have always to be linked to articles (see section 2.15). Thus, statements on properties of several property classes of an article can be made in one constraint.
- A constraint consists of up to four sections, each introduced by a keyword plus colon and ended by a point:
- **Objects:**

  This section names the objects, on which are made statements in the constraint. Objects mean in this case property classes and properties. Several object declarations are separated by commas.

  In the following constraint sections, property classes and properties are addressed via the name of variables, which have been declared for them in the Objects section:

  - Variables for property classes are declared with the help of the construct IS_A:

    <Variable> IS_A <Property class>

    If one of the declared property classes is not assigned to the article to which the constraint is linked, the constraint will not be evaluated.
  - Declarations of variables for properties follow the declaration of the class to which the properties belong. The property declarations are introduced by the keyword WHERE. Several property declarations have to be separated by a semicolon. A property declaration has the following form:

    <Variable> = <Property>

  If no proper variable is defined for a property, it can be addressed via the variable of its property class in the following constraint sections: <Property class variable>.<Property>.

- **Condition:**

  indicates the condition which has to be fulfilled, so that the constraint is evaluated.

  The general determinations for conditions according to the language definition `OCD_1` apply for the syntax of this section.

  If the result of the evaluation of the condition is undefined, the constraint will not be evaluated, because no conclusion is possible on whether it is allowed to be evaluated.

  This constraint section is optional. If it is absent, the constraint is evaluated.

- **Restrictions:**

  indicates the relations, that have to exist between the properties of the article, so that the current configuration of the article is considered as consistent. Several relations are separated from each other by commas.

  Without the `Inferences` section (see below) the relations simply represent conditions. If one of the relations is not fulfilled or no conclusion on that is possible (undefined logical expression), the constraint is not fulfilled and the article thus has an inconsistent (invalid) configuration. The runtime environment makes sure, that either an inconsistent article cannot be ordered or that a property modification, which would lead to an inconsistent state, must not be executed.

  Together with the `Inferences` section, the relations can simoultaneously effect value assignments or restrictions of value sets. In these cases, the constraint forces a consistent configuration.

  All relations can be equipped with a condition. This condition has to be indicated after the keyword `IF`. The relation will only be evaluated, if the condition is fulfilled.

  Relations can be described by:

  - *Value comparison*

    The expressions on both sides of the equality operator `=` have to be identical.
  - *Value set examination*: <`Property name`> `IN (`<`Value set`>`)`

    The value of the property indicated on the left side must be included in the value set indicated in the right side of the expression.
  - *Calling a value combination table*

    An entry corresponding to the current configuration of the article must be included in the value combination table.

    The syntax and semantics of the call of tables are described in the next section.

- **Inferences:**

  determines the properties (comma-separated listing) to which values are supposed to be assigned via the constraint or for which their value set is supposed to be restricted[53].

  This constraint section is optional. It is omitted, if no values are to be assigned or no value set to be restricted.

  The assignment or restriction is done with the help of the relations in the `Restrictions`–section.

  - *Value assignment via value comparison*:

    If an operand of the equality operator is a property variable and if the property is listed under `Inferences`, the relation effects an assignment of the value of the other operand to the property, provided that the value of the other operand is defined[54].

    Example:

    ```
    Objects:
       cup IS_A cupboard_a.
    Condition:
       cup.design_group = 'A'.
    ```

[53] Properties, whose value sets can be restricted by means of a constraint, have to be marked as restrictable in the property table.

[54] The comparison is thus always fulfilled.

```
Restrictions:
   cup.colour_door = 'F002'.
Inferences:
   cup.colour_door.
```

If the left operand is a property variable and the property is *not* restrictable and the value of the right operand is defined, an assignment takes place in any case, even if the property is not listed under `Inferences`.

– *Restriction of value set via* `IN`*–expression*:

If the property is listed on the left side of the `IN` expression under `Inferences` and is restrictable, the expression causes a restriction of the value set of the property to the value set indicated in the right side of the expression[55]. The quantity indicated in the right side of the expression must not comprise any interval values in this case.

Example:

```
Objects:
   cup IS_A cupboard_a.
Restrictions:
   cup.colour_corpus IN ('F001', 'F002', 'F003').
Inferences:
   cup.colour_corpus.
```

– *Value assignment and restriction of value set via table call*:

Example:

```
Objects:
   cup IS_A cupboard_a
   where
      design = design_group;
      corpus = colour_corpus.
Restrictions:
   TABLE colours_corpus (design_group = design,
                         colour_corpus = corpus).
Inferences:
   corpus.
```

Syntax and semanctics of the table call are described more fully in the next section.

A value set of a restrictable property, once restricted, cannot be re-extended through a subsequently evaluated constraint. That means, if the value set, which is defined by an `IN`–expression or a table call, includes a value which is not included in the current value set, the value is not taken over into the new value set.

B.2 Table call

With the aid of the function `TABLE()`, value combination tables can be accessed in preconditions, actions, reactions, post-reactions and constraints.

The general syntax of the call is:

`TABLE <Table name> (<Parameter list>)`

For the table name all alphanumeric characters including the underscore are allowed. The first character must not be numeric.

`<Parameter list>` is a comma-separated list of the access parameters:
`<Table property> = <Actual parameter>`.

[55] The `IN` relation is thus always fulfilled.

Actual parameters can be:

- numeric or character string constant
- property variable

Names of article properties and names of table properties do not have to be identical, however, the actual parameter has to be able to be compared with the value of the associated table property.

The semantics of the table call vary in the different relationship types and are described in the following subsections.

B.2.1 Table call in preconditions

In preconditions, the table call is a logical expression:

- If one of the actual parameters does not have a defined value in the current configuration, the table is not accessed and the expression is undefined. (For dealing with undefined logical expressions see appendix A).
- If exactly one entry for the actual parameters is found in the table, the result is *true*, otherwise *false*.

B.2.2 Table call in actions, reactions and post-reactions

Within actions, reactions and post-reactions, value combination tables are used to assign values to properties.

The properties, to which a value is to be assigned via a table call, have to be indicated in the parameter list as actual parameters and to be equipped with the prefix "`$SELF.`". All other actual parameters serve as access key. An exception is the use of specific variable `VARCOND`[56] in price relationships for setting variant conditions: If this variable is specified as an actual parameter, so to this variable always a value will be assigned through the table call.

If no or all actual parameters are equipped with the prefix "`$SELF.`", values are assigned to all those properties which are stated as actual parameters, but have no value at the time of the call.

There is no table access and thus no value assignement, if either no actual parameters could be defined as access key according to the above described rules, or if one of the actual parameters explicitly defined as key parameter has no value.

The table call has to yield a unique value for each property to be assigend. Otherwise the call fails and there will be no value assignement.

B.2.3 Table call in constraints

The handling of table calls within constraints is more complex than within actions and depends both on the current evaluation context concerning the actual parameters and on the `Inferences` section:

- If the `Inferences` section is missing or if all actual parameters have defined values, the table call solely serves to check the consistency:
  All actual parameters then serve as key to access the table.

  If one of the actual parameters does not have a defined value in the current configuration (and if `Inferences` section is missing), there will be no table access and the constraint is not fulfilled. If the table contains no entry for the key defined through the actual parameters, the constraint is not fulfilled either.

[56] resp. the variable which is stated in version information table to be used instead

- If one or several restrictable properties, which are passed as actual parameters in the table call, are not evaluated (do not have a defined value), the other actual parameters are used as keys for table access, except for non–restrictable properties, which are cited in the `Inferences` section. Values are then derived for those properties that are cited under `Inferences`:
  - With a restrictable property, the value set is limited to the intersection of the value set before the evaluation of the constraint and the value set that was delivered by the table access.
  - With a non–restrictable property, the value delivered by the table access is assigned to that property. If the table access has delivered several values, the constraint is not fulfilled.

  If a restrictable property specified as an actual parameter is not evaluated (does not have a defined value) and is not cited under `Inferences`, its value set remains unchanged.

- If all actual parameters are property variables and if all these properties are restrictable, but are not evaluated in the current configuration (do not have a defined value), no key exists for a table access.

  In this case, the table call is handled as follows:
  All logical lines of the table are read out in sequence. For every property to be derived (cited under `Inferences`), it is checked whether the values specified in the logical line for this property are contained in the currently limited value set. If the values in the logical table line for all properties to be derived are valid, the respective values are transferred to one list per property (whereby a multiple occurrence of values is excluded). After all logical table lines have been processed, the value sets thus determined are then assigned to the respective properties as new (limited) value sets.

C Language definition OCD_3

This language definition comprises all determinations from the language definition OCD_2. Additionally, the following further determinations regarding the processing of multivalued properties and regarding multilevel configuration are valid.

C.1 multivalued properties

- The logical expression

  SPECIFIED <multivalued property>

  is true, if at least one value is set.

- In IN comparisons, a multivalued property can be on the right side only. There has to be a constant on the left side.

  Example: 'ABS' IN Special equipment

- In normal comparisons, 2 multivalued properties must *not* be compared with each other. A multivalued property is only allowed to be compared with a constant.

  Example: Special equipment = 'ABS' is true, if the value ABS is set (regardless, whether other values are possibly set).

- In the Condition part of constraints, 2 multivalued properties can be compared with each other, too. The comparison is true, if both value sets are identical.

- In the assignment part of actions and in the restrictions part of constraints, multivalued properties are allowed on both sides of the expression. In this case, all currently set values of the property on the right side are taken over for the property on the left side.

C.2 multilevel configuration

In relations of articles which are or can be subitems of a composite article (see section 2.6), it is necessary to differentiate, whether a property of the currently configured article is referenced, or a property of a superordinate article. For that purpose, the following classifiers for properties are defined:

- $self:
  refers to the currently configured article
- $parent:
  refers to the direct superordinate article
- $root:
  refers to the highest article in a multilevel configuration

The classifiers precede the property name, separated by a point. If a property is not classified, $self is assumed.

D Language definition OCD_4

This language definition includes all determinations from the language definition OCD_3. Additionally, the following further determinations apply:

- Instruction block:

  In relations of types *Action*, *Reaction* and *Post–Reaction* instructions can be pooled using curly braces {...}.
  This is in particular useful when multiple instructions have to be associated with the same condition.

- Range of values in IN–expressions:

  In addition to discrete values also upward resp. downward open and closed ranges (intervals) can be specified in the value list.

  Operators for upper limit: < (or LT), <= (or LE)
  Operators for lower limit: =>[57] (or GE), > (or GT)

  downward open range: <upper limit op><discrete value>

  upward open range: <lower limit op><discrete value>

  closed range of values: <discrete value>-<discrete value>

  The values given for a closed range as the lower and upper limits are part of the range of values.

  example:

  ```
  width IN (5-10, 20, >30)
  ```

- Placeholder in IN–expressions:

  In comparisons with an IN–expression[58], the placeholder characters '*' and '?' can be used in string constants of the value set. The '*' character replaces any (0-n) number of characters and the '?' character replaces exactly one character of the comparison operand on the left-hand side.

  Examples:

  - Comparison with IN ('F*O') is true if the value of the left operand begins with the character 'F' and ends with the character 'O'. Any number of other characters (or none at all) can be between these.
  - Comparison with IN ('F?O') is true if the value of the left operand begins with the character 'F', ends with the character 'O' and exactly one other character is between these.

  However, placeholder substitution is only effective if it is activated via field *PlaceHolderOn* in the version information table (section 2.23). (Otherwise the two comparison examples above are only true if the left operand contains the exact character string 'F*O' resp. 'F?O'.)
  If the replacement characters ('*', '?') are also used as regular characters in values of string properties, then, when placeholders are activated, an exact comparison can be carried out with the aid of normal comparative operators, because placeholder substitution in general does not occur in those.

- Optional Objects section in Constraints:

  If the Objects section is missing, the identifiers used in the other sections of the constraint refer to the article's properties with the same name.

- Logical expressions in the Restrictions–section of Constraints:

  When comparing values in the Restrictions–section logical expressions are generally accepted (not only the equality operator as in the language definition OCD_3). The constraint fails if the condition defined by the expression is not definitly met.

[57] alternatively the form => may be used
[58] i.e., not, if the IN–expression is used in a constraint in order to restrict the value set

Properties referenced in the expression must not be stated in the (possibly existing) `Inferences`–section of the constraint!

- Output of messages to the user:

  In the relationship types *Action*, *Reaction* and *Post–Reaction*, the function

  `USER_MESSAGE(<Text Number>)`

  can be used. This function outputs the message to the user, which is stored in the text table `UserMessage` under the specified text number (ID). (The text number has to be specified as a string constant, i.e. enclosed in inverted commas.)
  This can be used to assist the user with complex object dependencies.
  The output uses the language that is currently set in the application. If no text is stored for this language in the table `UserMessage` under the specified text number, there is no output of a message.
  As with assignments, a condition can follow the call of the function `USER_MESSAGE()` after the keyword `IF`. This condition must be fulfilled for the message to be output.

- Abort of the evaluation of relationship knowledge:

  In relations of types *Reaction* (for a property) and *Post–Reaction* function

  `ABORT()`

  can be used to cancel the evaluation of relationship knowledge after a property change. (The configuration state of the article after the abort is then the same as before the attempted property change.)

  This can be useful/necessary with properties allowing the free entry of values by the user if not all values are allowed. Typically, the function is concatenated with an `IF`–expression, putting the condition which must be definitely fulfilled, so that the function is executed.

- Substring generation:

  The function `SUBSTR(<String>, <Position> [,<Length>])` results in a character string with contents that correspond to the substring of the string supplied in the first parameter, which begins at the position specified in the 2nd parameter and has the (maximal) length supplied in the 3rd parameter.

  The first parameter can be specified as an expression whose evaluation yields a character string. (The function itself can be used in string expressions, too.)

  The numbering of the character positions starts at 0. If the sum of position (2nd parameter) and length (parameter 3) results in a position that exceeds the length of the passed string, the substring is formed starting from the specified position to the end of the given string.

  The length parameter is optional. If it is not specified, the substring is formed starting from the specified position to the end of the given string.

  In the case of the following errors in the parameters, the function delivers an empty character string:

  - The first parameter is not a string.
  - The values for position and/or length parameters are not integral.
  - The values for position and/or length parameters are negative.
  - The position exceeds the length of the string.

- The following functions for processing strings are available:

  - Function `SIZE(<string>)` returns the number of characters contained in the string that is passed in the parameter.
  - Function `ToUPPER(<string>)` returns a string, with all lower case letters in the string that is passed in the parameter are converted to upper case.

- Function `ToLOWER(<string>)` returns a string, with all upper case letters in the string that is passed in the parameter are converted to lower case.
- Function `TRIM(<string>)` returns a string, in which all spaces located at the beginning and at the end of the string that is passed in the parameter are removed.
- Function `RTRIM(<string>)` returns a string, in which all spaces located at the end of the string that is passed in the parameter are removed.
- Function `LTRIM(<string>)` returns a string, in which all spaces located at the beginning of the string that is passed in the parameter are removed.

The string parameter can be specified as an expression whose evaluation yields a character string. (The functions themselves can be used in string expressions, too.)

- The following functions for converting strings into numbers are available:
  - Function `FLOAT(<expression>,<fallback>)` returns a fractional number depending on the type of the expression passed in the first parameter:
    - If the expression itself produces a fractional number, this is also the result of the function.
    - If the expression evaluates to an integer, the result is the corresponding fractional number, i.e. with decimal part `0`.
    - If the expression produces a string, an attempt is made to convert it into a fractional number. If successful, this is the result of the function, otherwise the fallback value specified in the second parameter. The same regulations apply to the representation of the number in the string as for the representation of fractional numbers in the property value table, i.e., a point (`'.'`) is used as the decimal separator, a thousands separator is *not* used, and in the first place can be a minus sign.

    If the 2nd parameter is not a fractional constant, a syntax error is triggered (and the evaluation of the relationship will be aborted).
  - Function `INT(<expression>,<fallback>)` returns an integer number depending on the type of the expression passed in the first parameter:
    - If the expression itself produces an integer number, this is also the result of the function.
    - If the expression evaluates to a fractional number with decimal part `0`, the result is the integral part.
    - If the expression evaluates to a fractional number with decimal part not equal `0`, the result is the fallback value specified in the second parameter.
    - If the expression produces a string, an attempt is made to convert it into a integer number. If successful, this is the result of the function, otherwise the fallback value specified in the second parameter. The same regulations apply to the representation of the number in the string as for the representation of integer numbers in the property value table, i.e., a thousands separator is *not* used and in the first place can be a minus sign.

    If the 2nd parameter is not a integer constant, a syntax error is triggered (and the evaluation of the relationship will be aborted).

- Visibility of properties:

  Function

  `SET_VISIBILITY(<property>, <logical expression>)`

  can be used in relations of types *Action*, *Reaction* and *Post-Reaction* in order to control the visibility of a property to the user of an OFML application:
  - If the evaluation of the logical expression returns `false`, the specified property is made invisible to the user.
  - If the evaluation of the logical expression returns `true`, the specified property is made visible to the user.

– If the evaluation of the logical expression returns an undefined result, the function call has no effect (i.e., the visibility state of the property does not change).

The function call can be equipped with a condition. This condition has to be indicated after the keyword `IF`. Then, the function call will only be executed if the condition is definitely fulfilled.

The visibility control via `SET_VISIBILITY()` is an alternative to the use of *preconditions*:
Function `SET_VISIBILITY()` only affects the visibility to the user of an OFML application (property editor, variant text). In technical contexts[59], however, the feature is still available (with the most recently assigned value).
A property hidden by preconditions, however, is completely invalid (invisible), i.e., is not available also in technical contexts.

Thus, one consequence of the semantics of preconditions is, that the call of `SET_VISIBILITY()` has no effect if the specified property is currently hidden by a precondition[60]! Therefore, a combination of preconditions and `SET_VISIBILITY()` for one and the same property is not useful and not recommended.

- Changing tax categories:

  By means of function

  `SET_TAX_CATEGORY(<tax type>, < tax category>)`

  in `TAX` relationships (see section 2.15) the tax category assigned to the article in its taxation scheme for given tax type can be changed (overwritten).

  Typically, such a change is linked with a condition specified by an `IF` construct[61]. (Then, the function call will only be executed if the condition is definitely fulfilled.)

  As identifiers for tax type and tax category the predefined identifiers from appendix H have to be used.

  `TAX` relationships can be linked with the following entities. They will be evaluated in the given order:

  1. article
  2. property classes of the article
  3. currently assigned properties of the article[62]
  4. values of currently assigned properties

  If the function is called for a given tax type in several relationships, the assignment of the respective last call applies.

[59] evaluation of relationship knowledge, graphical data, variant codes and other technical descriptions of an configuration
[60] I.e., when the property becomes valid again due to the preconditions, it has the same visibility state as at the time when it was hidden.
[61] An example would be the change of the material category in tax type `ECO_FR` depending on a specific article variant.
[62] including non-configurable properties

E Language definition SAP_LOVC

This language definition refers to the language which is used in the variant configuration of the SAP ERP (part of the logistics module) for coding relationship knowledge. By default, all language constructs are supported, which have the same syntax and semantics as the correspondent constructs of the OCD language definitions[63].

Accordingly, (currently) the following language elements and constructs are supported:

- relation types: *Precondition*, *Selection condition*, *Procedure*[64], *Constraint*
- Condition part: `IF`
- Logical operators: `AND`, `OR`, `NOT`
- Built-in conditions: `IN`, `SPECIFIED`
- Comparison operators: < or `LT`, <= or `LE`, = or `EQ`, <> or `NE`, >= or `GE`, > or `GT`
- Arithm. operators: `+`, `-`, `/`, `*`
- Arithm. functions: `sqrt()`, `abs()`, `sign()`, `frac()`, `trunc()`, `ceil()`, `floor()`
- Price factors: `SET_PRICING_FACTOR()`[65]
- Evaluation of variant tables: `TABLE()`

  The table call in OCD actions[66] (see section B.2.2) has the following special feature: As the evaluation alternatives specified in the SAP data base are unknown in OCD, it has to employ other ways in order to determine, which properties in the set of actual parameters serve as key for the table access, and to which properties a value is to be assigned via the table call. In OCD, this is done by using the prefix `$self` for the properties to which a value is to be assigned. Conversely this means that properties used as access key *may not* be qualified with `$self`[67]!

[63] Beyond that, providers of OCD implementations can support further constructs. As needed, these contructs have to be inquired from the provider of the relevant OFML application.
[64] is supported only to the extent, as defined for OCD relation type *Action*
[65] slightly different signature (see section 3.4)
[66] SAP: procedure
[67] In the variant configuration of SAP ERP for these properties then the qualifier `$root` is assumed.

F Arithmetic functions in relational knowledge

The following functions can be used in arithmetic expressions in the code of relations[68].

The arguments themselves can be arithmetic expressions. If an argument is an undefined arithmetic expression, the result of the function is also undefined.

In case of invalid arguments, the evaluation of the relation is aborted.

pow(x(Float), y(Int)) → Float
: The function *pow*() calculates x to the power y. If x is negative, y must be integral. If x is 0, y must be positive. The result is 1.0, if both x and y are 0.

sqrt(x(Float)) → Float
: The function *sqrt*() calculates the not negative square root of x. x must not be negative.

fabs(x(Float)) → Float
: The function $fabs()$ calculates the absolute value of x.

ceil(x(Float)) → Float
: The function *ceil*() calculates the smallest integral value, which is not smaller than x.

floor(x(Float)) → Float
: The function $floor()$ calculates the biggest integral value, which is not bigger than x.

sign(x(Float)) → Int
: The function *sign*() provides the algebraic sign (`-1` or `+1`) of x.

trunc(x(Float)) → Float
: The function *trunc*() provides the integral part of x.

frac(x(Float)) → Float
: The function $frac()$ provides the decimal part of x.

[68] Except for the functions `sign`, `trunc` and `frac`, all functions also belong to the arithmetic standard functions of OFML.

G Reserved keywords

Below for each language definition the reserved keywords are listed.

These may not be used as property identifiers when using the respective language definition![69]

Please note:
Below, the keywords are written in upper case. However, since OCD relationships are generally case insensitive (see A), mixed notations as well may not be used as property identifiers.

language definition **OCD_1**:

- `AND`
- `BAN`
- `EQ`
- `GE`
- `GT`
- `IF`
- `IN`
- `LE`
- `LT`
- `NOT`
- `OR`
- `SET_PCKG_FACTOR`
- `SET_PRICING_FACTOR`
- `SPECIFIED`

language definition **OCD_2**:

all keywords from `OCD_1` plus:

- `CONDITION`
- `FALSE`
- `INFERENCES`
- `IS_A`
- `OBJECTS`
- `RESTRICTIONS`
- `TABLE`
- `WHERE`

language definition **OCD_3**: as per `OCD_2`

language definition **OCD_4**:

all keywords from `OCD_3` plus:

- `ABORT`
- `USER_MESSAGE`

language definition **SAP_LOVC**:

all keywords from `OCD_2` (except `BAN` and `SET_PCKG_FACTOR`) plus all keywords according to the SAP ERP specification

[69] In addition, software providers of OCD implementations may define further restrictions. These are to be requested from the respective software provider.

H Tax Types and Tax Categories

This table contains the standardised — within OCD — tax types and associated tax categories[70].

Tax Type (Explanation)	Tax Category	Explanation
VAT (value added tax)	standard_rate	standard rate
	reduced_rate	reduced rate
	super_reduced_rate	severely reduced rate
	parking_rate	parking rate
	services	service
	zero_rate	zero rate
	exemption	tax-excempt
ECO_FR (ECO tax France)	seat_metal	seat, mostly consisting of metal
	seat_metal95	seat, consisting of more than 95% of metal
	seat_wood	seat, mostly consisting of wood
	seat_plastics	seat, mostly consisting of plastics
	seat_other	seat, other materials
	storage_metal	storage, mostly consisting of metal
	storage_metal95	storage, consisting of more than 95% of metal
	storage_wood	storage, mostly consisting of wood
	storage_plastics	storage, mostly consisting of plastics
	storage_other	storage, other materials
	workplace_metal	workplace, mostly consisting of metal
	workplace_metal95	workplace, consisting of more than 95% of metal
	workplace_wood	workplace, mostly consisting of wood
	workplace_plastics	workplace, mostly consisting of plastics
	workplace_other	workplace, other materials
	other_metal	other furniture, mostly consisting of metal
	other_metal95	other furniture, consisting of more than 95% of metal
	other_wood	other furniture, mostly consisting of wood
	other_plastics	other furniture, mostly consisting of plastics
	other_other	other furniture, other materials
	none	non-taxable

Notes regarding ECO tax France[71]:

- The Non-profit company Valdelia[72] is authorized by the French government to implement the provisions relating to Eco tax in the B2B trade of office furniture[73]. The purpose of the tax is the involvement of customers in the disposal and recycling of bulky furniture.
- The tax depends on the net weight of the article and its tax category, which is associated with a price factor. Net weight multiplied by this factor produces the tax amount. The price factors vary annually and are announced in October. The tax is the first time raised since 1st March 2013.
- The tax categories mentioned above correspond to the tax categories stated in the document "Barème eco-contribution"[74]. The tax category combines the type of furniture and the material category[75]. The assignment to a material category stems from the weight. For example, if wooden materials make more than 50 percent of the weight of an article, it is assigned to material category *wood*. If no material type has the majority of the weight the article has to be assigned to material category *other*.

[70] Requests to record other tax types and tax categories should be made to the OFML Standardisation Committee.
[71] The VAT will be charged on the Eco tax and the tax is not discountable.
[72] www.valdelia.org
[73] For the recycling of furniture intended for private customers (B2C trade), organization éco-mobilier is responsible. Their control model is not (yet) subject of OFML standardization.
[74] issued by Valdelia
[75] The material category `metal95` is valid from July 1, 2014.

I Terms

- **EAN**
  - Abbreviation for European Article Numbering
  - Non-profit organization founded in 1977 with the objective to standardize identifications of products, units, systems, etc. for efficient execution of business processes in trade.
  - Since 1992, after members from other continents have joined the association, active as EAN International.
  - In cooperation with the Uniform Code Council (UCC)[76], the corresponding authority for North America, the EAN.UCC system to identify products (→ GTIN), locations (→ ILN), etc. was developed.
- **GTIN**
  - Abbreviation for Global Trade Item Number
  - Unique identification number assigned in the context of the → EAN.UCC system for an item (product or service), which can be ordered and charged within business processes in trade.
  - Several code schemes can be used within the EAN.UCC system (EAN.UCC-14, EAN.UCC-13, EAN.UCC-8).
- **GLN**
  - Abbreviation for Global Location Number
  - Unique identification number (13-digit) assigned in the context of the → EAN.UCC systems for physical and electronical addresses of companies, affiliated companies, branch offices as well as organisationally relevant operating units.
- **Intrastat**
  - Intrastat are statistics led by the Federal Bureau of Statistics in Wiesbaden, which comprises the inner-European trade with Germany.
  - Every German company trading within Europe has to report this according to exactly determined standards. This report is to be done monthly.
  - Every trade item has an 8-digit number, which is listed in the Statistical and Tariff Classification for International Trade and has to be indicated together with weight, value, transport route, etc.
- **Tariff code**
  - Tariffs codes are used in business transactions with countries that do not belong to the European Union.
  - From a goods value of 1.000 euros upwards, a German exporter has to fill in a written export notification for the customs authorities and the Federal Bureau of Statistics.
  - A goods tariff code is necessary for the declaration of each trade item. In order to allow the allocation of the items, a precise declaration of the items according to the *Statistical and Tariff Classification for International Trade* is necessary.

[76] since 2005 operating under the name *GS1* (Global Standards One)

J Modification history

J.1 OCD 4.3 vs. OCD 4.2

- Correction regarding multi-valued properties in section 2.9.
- Introduction of *Property groups* (section 2.11).
- Post-reactions now can also be specified for articles (section 2.15).
- Using `TAX` relationships tax categories now can be defined for specific article variants (sections 2.15 and 2.25).
- Removal of length limitation for lines in the text tables (section 2.20).
- Removed alternative text tables (section 2.20).
- Language definition `OCD_2` now also defines boolean constant `TRUE` (appendix B).
- Language definition `OCD_4` contains new function `SET_VISIBILITY` to be used to control the visibility of a property to the user of the OFML application as well as new function `SET_TAX_CATEGORY` to assign a tax category deviating from the taxation scheme (appendix D).
- The language definitions `SAP_3_1` and `SAP_4_6` now are combined into `SAP_LOVC`.