Specification

OCD
OFML Commercial Data*(OFML Part IV)

Version 3.0

Status: Release

Thomas Gerth, EasternGraphics GmbH (Author and Editor)

2005-05-31

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
- Support for statistics and delivery control

OCD is *no* format to create catalog data. This has to be made available in a different and superordinate way. The connection between catalog and product data is effected via the respective software system with the help of the article number.

The data modell for OCD is based on the fundamental OFML product data model (see annex 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 set. Blank lines (consisting of zero or several blank characters or tabulator) are ignored.
- The fields of a data set 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 set 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 absolutely filled in (obligatory field)

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.

[1] There are no restrictions for CSV data sets conerning the individual field lengths, however, for certain fields of the data type `Char`, maximum possible or senseful lengths resulting from the purpose are indicated.

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 as well as decimal point, poss. minus sign, in the first place

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)

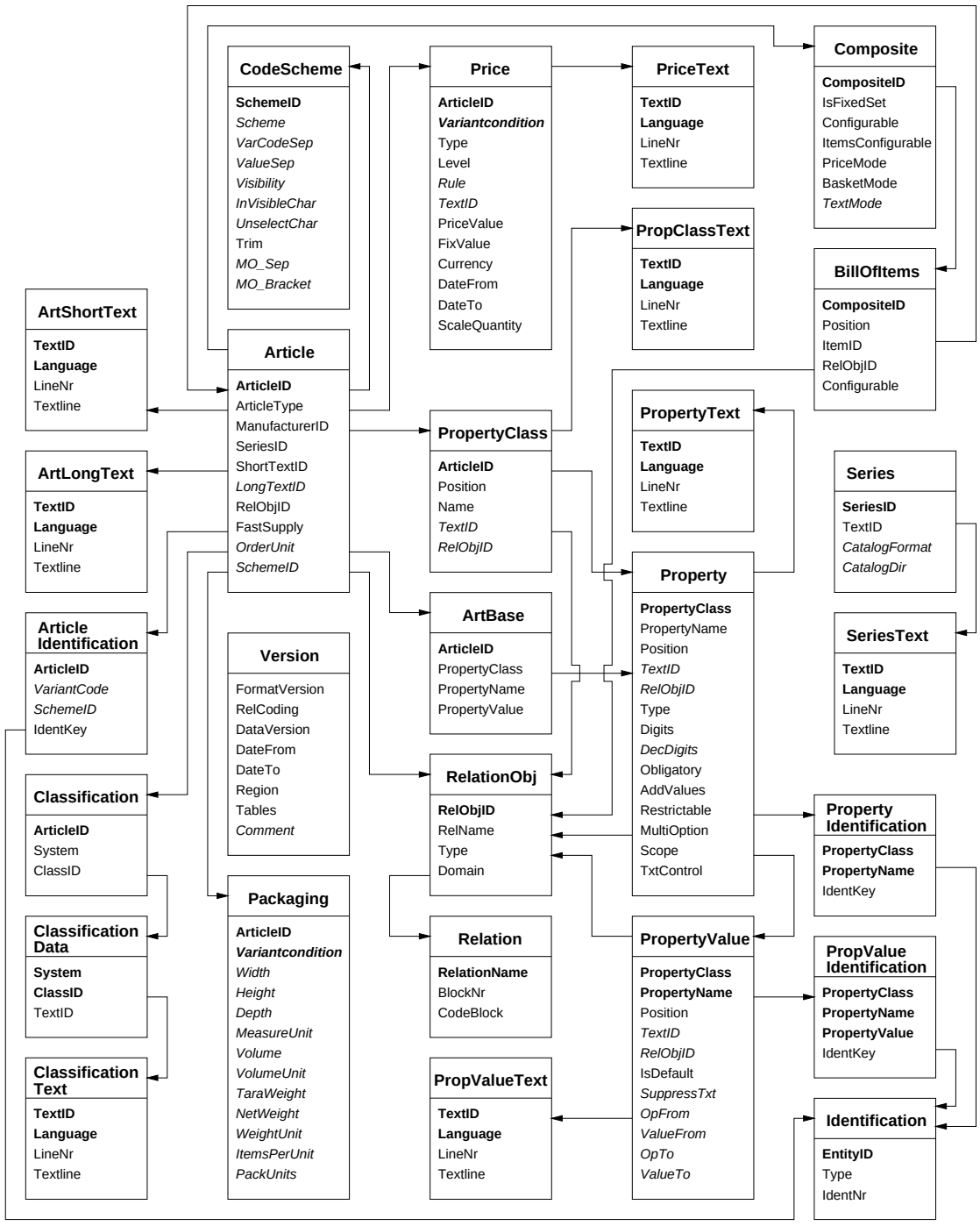
The obligatory field mark is only relevant for fields of the data type *Char*. There has always an indication to be made for fields of the other data types. Concerning the fields of the data type *Num*, the respectively possible values can be found in the description of the respective tables.

Fields for *units* are defined in different 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



Key fields are highlighted by bold print and the fields, which are no obligatory fields, by italic print.
The alternative text tables (see par. 2.18) are not presented for the sake of clarity.

2.2 The article table

Table name: `Article`
Obligatory table: yes

No.	Name	Key	Type	Length	Obligation	Explanation
1.	ArticleID	X	Char		X	Base article number
2.	ArticleType		Char		X	Article type
3.	ManufacturerID		Char		X	Manufacturer abbreviation
4.	SeriesID		Char		X	Series abbreviation
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.	OrderUnit		Char	3		Order unit
10.	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 par. 2.3).

- The article type (field 2) determines fundamental properties 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 text numbers (fields 5 and 6) serve as key for the tables with the short resp. long descriptions of the articles (see par. 2.18).

- The relational object number serves as key to access the relational object in the relational object table (see par. 2.14), to which the price relations to derive variant conditions for the article are linked (see par. 3). (If no relational object is needed, the number `0` has to be indicated.)

- 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.

[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.

- Field 9 indicates 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` – *item*, `MTR` – *meter* and `MTK` – *square meter*. If no indication is given, *item* is used as default unit.

- The identifier indicated in the 10th field serves to reference the codification scheme from the table `CodeScheme` (par. 2.22), 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 final article number will be generated for the article.

2.3 The article identification table

Table name: `ArticleIdentification`
Obligatory table: nein

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

Remarks:

- The table serve to indicate additional identification numbers (of different types) for articles. The additional identification numbers are not entered directly into this table, but indirectly via the key indicated in field 4 in the table `Identification` (see par. 2.20).

- 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 number scheme table, see par. 2.22).

- If there are several entries for a base article in the table, a variant code has to be indicated for each entry. If no correspondign variant code is found to a given configuration of an article, no identification of the desired type can be determined for this article.

2.4 The classification table

Table name: `Classification`
Obligatory table: no

No.	Name	Key	Type	Length	Obligation	Explanation
1.	ArticleID	X	Char	18	X	Base article number
2.	System		Char	80	X	Name of the classification system incl. version
3.	ClassID		Char	80	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. Presently, the following systems are allowed[4]:

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 abbreviation, 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: non

No.	Name	Key	Type	Length	Obligation	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[5].
- The text number serves as key for the table `ClassificationText` (see par. 2.18), in which language-specific texts describing the classification can be stored.

[4] An extension is planned for a later version, which also allows a classification concerning *Recycling properties*.
[5] 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	Obligation	Explanation
1.	ArticleID	X	Char		X	Base article number
2.	Variantcondition	X	Char		X	Variant condition
3.	Width		Num			Width of the packaging unit
4.	Height		Num			Height of the packaging unit
5.	Depth		Num			Depth of the packaging unit
6.	MeasureUnit		Char	3		Unit of measurement of the dimensions 3 to 5
7.	Volume		Num			Volume of the packaging unit
8.	VolumeUnit		Char	3		Unit of measurement of the volume
9.	TaraWeight		Num			Weight of the packaging unit
10.	NetWeight		Num			Weight of the individual article in the basic design
11.	WeightUnit		Char	3		Unit of measurement of the weights 9 to 10
12.	ItemsPerUnit		Num			Number of the articles per packaging unit
13.	PackUnits		Num			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[6].
- A packaging unit can include several articles (of the same number)(field 12). Components of the article, e.g. optional accessories, can also be delivered in separate packaging units (field 13).
- Depending on special properties, dimensions, volumes, weights and numbers of packaging units, which deviate from the basic design of the article, can be entered with the help of *variant conditions* (field 2). The use and handling of such variant conditions is described in par. 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.
- The following units of measurement are allowed in the fields 6, 8 and 11[7]:

  Lengths:

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

[6] An extension is planned for a later version, which also allows to make indications concerning the packaging of articles that are delivered in individual components, also combined, if applicable.

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

Volume:

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

Weight:

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

- 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 8) plus the product of the weight of the individual article (field 10) and the number of articles per packaging unit (field 12).

2.6 The table of composite articles

Table name: `Composite`
Obligatory table: no[8]

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 of functional aspects.

No.	Name	Key	Type	L"ength	Obligation	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 properties of a composite article. The sub items are determined in the bill of items (see next paragraph).

- 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 conditions of existence to be indicated for the sub items in the bill of items.
  If *yes* is indicated, possibly indicated conditions of existence for sub items will not be evaluated!

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

- 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 par. 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 the 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 acquisition. 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 relational knowledge (see below) that the value quantities 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:

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 the basket mode "T" are supposed to be described in the description text of the composite article[9]:

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

[9]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.

2.7 The bill of items

Table name: `BillOfItems`
Obligatory table: no

No.	Name	Key	Type	Length	Obligation	Explanation
1.	CompositeID	X	Char		X	Article number of the composite article
2.	Position		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 ?

Remarks:

- This table indicates, which sub items can be included in a composite article (see previous par.).
- The position of a sub item within the composite article (field 2) is taken into account in the order list output.
- An existence condition for the sub itme can be indicated via the relational object (field 4)(see par. 2.14). Existence conditions have to be indicated as relation type *pre-condition* with the scope "BOI". If an existence condition is entered and not fulfilled, the sub item will not be included in the current bill of items of the composite article.
  Generally, existence conditions are only evaluated, if *no* is indicated for the composite article in the field *IsFixedSet* of the table `Composite` (see previous par.)
- Field 5 indicates, whether the sub item is configurable. If *no* is indicated, it cannot be configured, even if property classes (see par. 2.8) are assigned to it. Generally, sub items are only configurable, if this is released through the composite article, see field *ItemsConfigurable* in the table `Composite`.

2.8 The property class table

Table name: `PropertyClass`
Obligatory: yes[10]

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

Remarks:

- In this table, the property classes, which describe the properties of the article, are assigned to the articles.

[10]The table can be omitted, if the database is not meant to store configuration data, but e.g. only article texts and prices.

- The position of the class within the quantity of the property classes of an article influences the order in enumerations of the properties of the article (listings, property editors, and others).
- Relations of the type *action* can be bound to the property class via the relational object (field 5)(see par. 2.14). (If no relational object is needed, the number `0` has to be indicated.)

2.9 The property table

Table name: `Property`
Obligatory table: yes[11]

No.	Name	Key	Type	Length	Obligation	Explanation
1.	PropertyClass	X	Char		X	Designator of the property class
2.	PropertyName		Char		X	Designator 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 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	Entry obligatory?
10.	AddValues		Bool	1	X	Additional values allowed?
11.	Restrictable		Bool	1	X	Value quantity restrictable?
12.	MultiOption		Bool	1	X	Multivalued property?
13.	Scope		Char	2	X	Scope, see table below
14.	TxtControl		Num		X	Text control

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

Remarks:

- In this table, the properties of each property class are listed.
- The names of the properties are symbolic (language independent) designators. Alphanumeric characters can be used, including the underscore, but this first character must not be a numeric character.
  Speaking (language dependent) designators (for the use in the user interfaces) are stored in the table `PropertyText` (see par. 2.18). 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).

[11] The table can be omitted, if the database is not meant to store configuration data, but e.g. only article texts and prices.

- The relational object number serves as key to access the relational object in the table `RelationObj` (see par. 2.14), 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) 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.
  - Values of the data types 'N' and 'L' are real or full numbers represented in a simple decimal point notation. Field 7 determines the maximum number of all digits without decimal point and possible minus sign, and field 8 the number used thereof for the presentation of the fractional part.
  - 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, paragraph "Format specifications for properties" in the annex "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 use the unit of measurement set by the user to represent resp. enter the value. The property editor therefore effects a convertion between the user-defined unit of measurement and the unit of measurement (m) used in OFML for units of length. 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 be evaluated by the user. As long as an article has obligatory properties, which are not evaluated, its configuration is not complete (invalid).
  This feature is only relevant for properties of the type 'C', numeric properties always require an entry, because otherwise, the system cannot execute any operations with such properties. If a value quantity (see table `PropertyValue`, par. 2.12) is preset for a property marked as optional, the application system automatically generates and uses additionally the pseudo value `VOID` for the state "not selected".
  The application system guarantees for obligatory properties with a preset quantity of property values and without the possibility of a free value entry (see field 10), that a value from the value table is always selected; such a property is thus always evaluated. For other properties, the completeness of the configuration has always to be assured by making available product relational knowledge (selection conditions, see table `RelationObj`, par.2.14).
  When using this feature, one has to observe, that it refers to the entry 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 quantity of values (choice list). Options from a product-logical point of view can also be displayed via OCD properties with obligatory feature. This is even obligatory for numeric properties (see above), the state "not selected" can and has to be represented by a special value (e.g. `0`).

- The feature in field 10 indicates, whether the user can enter values freely, if applicable in addtion to the values stored for the property in the value table. This can be use to enter free texts, quantities or measures. For numeric properties, it is recommendable to indicate an allowed interval in the value table (see table `PropertyValue`, par. 2.12).

- If the value quantity of the property is supposed to be restrictable with the help of relations of the type *Constraint* (see par.2.14), the feature in field 11 has to be set to `1`.

The quantity of the values of a normal *non*-restrictable property, from which the user can choose, comprises all values which are set for the property in the property value table **and** which either have no pre-conditions or whose pre- conditions are fulfilled with regard to the current configuration of the article. Obligatory properties have always one of the values.
Restrictable properties are handled in a different way in this aspect. The value quantity of these properties is restricted by contraints, according to the quantity of the values in the property value table. They are considered to be evaluated only when the value quantity 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.

- Multivalued 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 the scope "C" (or blank characters, i.e. no indication) are configurable. They are per se also visible and can be used for the generation of the graphical representation.
  - Properties of all other scopes of application are helping properties, which can be used within relational knowledge.
  - Furthermorde, properties of the scope "RV" are shown to the user as read-only and can also be used for the generation of the graphical representation.
  - Properties of the scope "RG" are not visible for the user, but they are needed for the generation of the graphical representation.
- Field 14 includes 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 paragraph 5. (The code `0` marks the standard procedure for single line texts.)

2.10 The property identification table

Table name: `PropertyIdentification`
Obligatory table: no

No.	Name	Key	Type	Length	Obligation	Explanation
1.	PropertyClass	X	Char		X	Designator of the property class
2.	PropertyName	X	Char		X	Designator 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 via the key indicated in field 4 in the table `Identification` (par. 2.20).

2.11 The article base table

Table name: `ArtBase`
Obligatory table: no

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

Remarks:

- This table describes the "fixed" properties of an article by assigning one or several values (subsequent data sets) to a property of the article.
- 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 par. 2.12)!

2.12 The property value table

Table name: `PropertyValue`
Obligatory table: yes[12]

No.	Name	Key	Type	Length	Obligation	Explanation
1.	PropertyClass	X	Char		X	Designator of the property class
2.	PropertyName	X	Char		X	Designator of the property
3.	Position		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

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) designators. Speaking (language dependent) designators (for a use in the user interfaces) are stored in the table `PropValueText` (see par. 2.18). For this purpose, a text number is assigned in the 4th field as access key. If no text number is defined or no text entered in a required language, the symbolic (language independent) designator (from this table) is shown for properties of the 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 par. 2.14), to which the relational knowledge for the property value is bound. (If no relational object is required, the number `0` has to be indicated.)
- If none of the values of a property is marked as proposal value (field 6), the first value is used as initial value for obligatory properties, for optional values the virtual value "not selected". For the last one, the internal abbreviation `VOID` is ... and reserved for properties of the type 'C'. 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.

[12]The table can be omitted, if the database is not meant to store configuration data, but e.g. only article texts and prices.

- The property value is entered in the fields 8 to 11. It is possible to indicate an interval for the property value. (This can be used in particular for measure properties.)
  - A fixed property value (without input area) is marked by the operator 'EQ'. It does not play a role, 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 area 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 area 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 values for the property, these values are also included into the choice list of the property generated for the 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. The upper limit could be set as default value for a closed interval.
  - There can only be one interval displayed for each property If several intervals are stored for one property, they must be equipped with pre-conditions excluding each other (see par. 2.14).
- Numeric values have to be indicated according to the number of digits resp. decimal digits specified for the property in the property table (see par. 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'

2.13 The property value identification table

Table name: `PropValueIdentification`
Obligatory table: no

No.	Name	Key	Type	Length	Obligation	Explanation
1.	PropertyClass	X	Char		X	Designator of the property class
2.	PropertyName	X	Char		X	Designator 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 via the key indicated in field 4 in the table `Identification` (par. 2.20).
- Additional identification numbers can only be indicated for values, which are also in the property value table (see par. 2.20) and which are no interval values[13].

[13] 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.

2.14 The relational object table

Table name: `RelationObj`
Obligatory table: yes[14]

No.	Name	Key	Type	Length	Obligation	Explanation
1.	RelObjID	X	Num		X	Relational object number (bigger `0`)
2.	RelName		Char		X	Relation name
3.	Type		Char	1	X	Type of the relation: 1 - Pre-condition 2 - Selection condition 3 - Action 4 - Constraint
4.	Domain		Char	4	X	Scope: C - Configuration P - Price relation BOI - Bill of items PCKG - Packaging relation

Remarks:

- This table pools relations to relational objects.
- Presently, the following relation types are possible:
  - *Pre-conditions* determine, whether a property is allowed to be evaluated, wheter a property value is allowed to be set or a component of the bill of items is allowed to be used. If several pre-conditions are indicated for one entity, the entity is only used, if all conditions are fulfilled.
  - *Selection conditions* determine, if a property has to be evaluated. When the consistency is checked during the order list generation, respective error messages are displayed for articles with not evaluated (optional) properties, whose selection conditions are fulfilled, (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 derive property values.
    Actions on articles and property classes are used in each configuration step. Actions on properties are used, if the properties are not hidden through a pre-condition. Actions on property values are used, 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. Values can also be derived or value quantities can be restricted. Constraints have to be bound to articles and are used in each configuration step.
    Not every language, that can be used to code relational knowledge, supports constraints (see also paragraphs 2.15 and 2.21).

[14]The table can be omitted, if no relational knowledge is needed.

- The scope (field 4) 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 in the price determination (see par. 3).
  Attention: since these relations are not necessarily evaluated together with the configuration relations, these relations must not include any value derivations for properties, which are also used in configuration relations.

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

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

  Price and packaging relations can only be of the type *action*.

2.15 The relational knowledge table

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

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

Remarks:

- The "knowledge" (the logic) about relations is stored in this table. Different languages can be used, whose syntax and semantics are described in the annex. The used language has to be indicated in the version information table (par. 2.21).
- Before the evaluation, the code blocks belonging to a relation are put together to a whole code block according to their number.

[15]This table can be omitted, if no relational knowledge is needed.

2.16 The price table

Table name: `Price`
Obligatory table: yes[16]

No.	Name	Key	Type	Length	Obligation	Explanation
1.	ArticleID	X	Char		X	(Base) article number
2.	Variantcondition	X	Char			Variant condition
3.	Type		Char	2	(X)	Type of price: GS - Gross sales price NS - Net sales price P - Purchase price
4.	Level		Char	1	X	Price level: B - Base price X - eXtra charge price TX1 - TaX 1 TX2 - TaX 2 D - Discount
5.	Rule		Char			Calculation rule
6.	TextID		Char			Text number
7.	PriceValue		Num		X	Price/Amount/Tax rate
8.	FixValue		Bool	1	X	Fixed amount (vs. percentage) ?
9.	Currency		Char	3	(X)	Currency of the fixed amount
10.	DateFrom		Date	8	X	Valid from
11.	DateTo		Date	8	X	Valid to
12.	ScaleQuantity		Num		X	Scale price–Quantity unit

Remarks:

- The base and striking prices as well as possible discounts and the tax rates to be used (each optional) for each article are registered in this table. The procedure of the price determination is described more fully in par. 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`).
- Both a sales price (either GS - discountable or NS – undiscountable) and a purchase price (field 3) can be indicated for each price item.
- The fields 2 and 3 are of no significance for entries which specify tax levels (level 'TX1' and 'TX2', field 4): for a (base) article, only one such tax level can be indicated. Alternatively, an article-wide tax level can be indicated by making an entry for the joker article "*" (field 1).
  If no specific tax rate and no article-wide tax rate are indicated for a given article, a tax rate fixed or set in the application system is used.

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

- The calculation rule modifies the manner of use of the price item when determining the price:
  - No special calculation rules are currently supported for base and striking prices. The amount indicated in field 7 is always added to the already accumulated price during the price determination. Striking prices can also be indicated as percentage values. In this case, the absolute amount results from the respective percentage of the base price.
  - The calculation rules '1' and '2' are presently supported for discounts. They indicate, whether the disount, if indicated in percent, is supposed to be calculated in relation to the base price ('1') or in relation to the price already accumulated during the price determination ('2').
  - The rule 'T' can be used for tax rates. This rule indicates that the tax(es) is (are) already included in the fixed amounts of base and striking prices.
- Via the text number (field 6), an explanation concerning the price entry can be indicated in the price text table, e.g. reason for extra charge price, description of the tax, etc. If there is no description for a price entry, 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`).
- 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 from a certain amount of ordered articles. Therefore, the amount is indicated in field 12, from which on the price from the table entry is meant to be used. By default, `1` has to be indicated here (no scale price).
  Attention: currently, scale prices are always calculated per order resp. contract position, but not throughout the whole order.

2.17 The series table

Table name: `Series`
Obligatory table: no

No.	Name	Key	Type	Length	Obligation	Explanation
1.	SeriesID	X	Char		X	Series abbreviation
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 system *DSR* of the company EasternGraphics for the registration of commercial forms.
- The text number in field 2 serves as key for the description table `SeriesText` (see par. 2.18). 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 includes 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)

2.18 The description tables

All text tables

Article short descriptions: `ArtShortText` (Obligatory table)
Article long descriptions: `ArtLongText`
Property class designations: `PropClassText`
Property designations: `PropertyText` (Obligatory table)
Property value designations: `PropValueText`
Price texts (Explanations concerning price components): `PriceText`
Series designations: `SeriesText`
Designations for classifications: `ClassificationText`

have the same structure:

No.	Name	Key	Type	Length	Obligation	Explanation
1.	TextID	X	Char		X	Text number
2.	Language	X	Char	2	X	Language
3.	LineNr		Num		X	Line number
4.	Textline		Char	80	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)[17] .
- A text consists of one or several lines of 80 characters each at the most. Presently, several lines are taken into account only in article long texts and property value designations[18].

As an alternative to the above described table definition, the below described extended table definition can be used for each text table. The number `2` has then to be added to the table name.

No.	Name	Key	Typ	Length	Obligation	Explanation
1.	TextID	X	Char		X	Text number
2.	Language	X	Char	2	X	Language
3.	Scope	X	Char	1	X	Scope M - Documents for the manufacturer C - Documents for customers
4.	LineNr		Num		X	Line number
5.	Textline		Char	80	X	Text line

[17]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 execute a conversion.

[18]in relation with the field *TxtControl* in the property value table (see par. 2.12)

Remarks:

- In addition to the language, the scope also serves as access key in this table definition. This allows to realize manufacturer- and trade-specific texts[19].
- Whether the scope is used for the determination of texts and in what context which scope is queried, depends on the respective application.
- Applications which support the scope in texts, first access the table with the ending `2`. If this table is not availabe or does not include any entry for the required scope, a second access is effected to the simple text table (without scope).

2.19 Value combination tables

Different languages to code relational knowledge (see par. 2.15) allow the use of value combination tables. In relational knowledge, value combination tables are used to check the consistency of a value combination, to derive values or to restrict the value scope of a property.

All possible value combinations are indicated via a defined quantity of properties in a value combination table.

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 designator, under which the table is called in relational knowledge, the addition "`_tbl`" and the suffix "`.csv`", the table name is entirely written in small letters.

Ex.:

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

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

No.	Name	Key	Type	Length	Obligation	Explanation
1.	LineNr		Num		X	Number of the table line
2.	PropertyName		Char		X	Name of the property
3.	Value		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[20].

Both property names and property values have to be written entirely in capital letters.

[19] In future version, the document type will also be allowed to be used as key.

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

Ex.:
The (logical) value combination table from the example above has been 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 a single property which refers to a restrictable product property (see par. 2.9), a value quantity can also be indicated for this property in the field of a table line.

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
```

2.20 The identification table

Table name: `Identification`
Obligatory table: no

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

Remarks:

- The table serves to indicate 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 annex)[21]:

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

Type	Explanation
CustomID	Dealer resp. big customer-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
ILN-1	International location number, type 1
ILN-2	International location number, type 2
Intrastat	Intrastat number
CustomsTarif	Tariff number

The type *CustomID* is used, if the data stock is designed for a specialist dealer or big customer, 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.

2.21 The version information table

Table name: `Version`
Obligatory table: yes

No.	Name	Key	Type	Length	Obligation	Explanation
1.	FormatVersion		Char		X	Number of the used OCD format version
2.	RelCoding		Char		X	used language for relational 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.	Tables		Char		X	included tables
8.	Comment		Char			free comments, additional information

Remarks:

- The table serves to indicate information concerning the used format and the product database. A version control system or other system can thus determine the structure and the usability of the database.
- The OCD format version (field 1) has to be indicated in the form `MajorNumber.MinorNumber` according to the OCD format specification.
- The language to code relational knowledge has to be specified in field 2 (see par. 2.15). The following languages can be used: `OCD_1`, `OCD_2`, `SAP_3_1`, `SAP_4_6`. The description of the languages can be found in the annex.
- 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 strictly monotonously ascendingly.
- The designator for the sales region[22] (field 6) can be freely assigned. However, it has to correspond to the designator, with which further required data (geometry, catalog) concerning the sales region is referenced in the respective software system.

[22]big customers with specific price lists or deviating configuration data are also represented by the *sales region* concept within this context.

- The tables, currently contained in the database, are listed in field 7, 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)[23].
  Additional blank characters after the commas are allowed.

2.22 The code scheme table

Table name: `CodeScheme`
Obligatory table: no

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 characteristics of the configurable properties are coded.
- The codification scheme, which is to be used for an article, is determined in the article table (par. 2.2) with the help of the scheme identifier[24].
- A difference is made between *pre-defined* and *user-defined* codification schemes. The respective codification procedures are fully described in the par. 4.
- The fields 3 to 10 serve to parameterize the codification schemes as described in the par. 4.

[23] that means without the prefix `ocd_` and without the suffix `.csv`.
[24] Principally, an individual codification scheme can thus be defined for each article.

No.	Name	Key	Type	Length	Obligation	Explanation
1.	SchemeID	X	Char		X	obvious identifier to reference the scheme
2.	Scheme		Char			Descriptionn 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 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		Character for brackets for multivalued properties

3 The price determination

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

3.1 Overview

The relevant price components (data sets in the price table) of the different price levels (field 4) are determined according to the following order:

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

The total price for the article is accumulated according to the calculation rule (field 5 in the price table) for each determined (valid) price component.

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

1. Determination of the price components for the article without variant condition.
2. Evaluation of all relevant price relations and determination of the price components for the article with the variant conditions, which are derived in these price relations. Price relations are marked by the type of use 'P' in the table `RelationObj` (field 4) and have to be of the type action ('3') (field 3): Variant conditions are derived in price relations (table `Relation`) by assigning the designation of the variant condition to the special variable `$VARCOND` (OCD language sets) resp. to the helping property `$self.variant_condition` (SAP language sets). The relevant price relations are determined according to the mentioned order from the relational objects
   1. of the article
   2. of the currently evaluated properties of the article
   3. of the current property values

   .

At the end, the taxes for the article are determined (entries in the price table with level 'TX1' resp. 'TX2'). If there is no specific tax rate and no article-wide tax rate for the article, a tax rate is used, which is fixed or set on the application system. Unless the rule 'T' is indicated, the such determined tax rate is charged up against the above determined net price. If several tax rates are relevant, the respective tax rates are separately charged up against the net price each independently of from the other.
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 price components

The relevant entry from the entries, which are read out of a price table, for a price level ('B','X','D') for an article without variant condition resp. with a concrete variant condition, is determined as follows:

1. 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 current date, are ignored.
2. 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 selection.
3. Those entries whose quantity indication for a scale price (field 12) is smaller than the quantity of the current order or contract position, are exluded from the remaining entries.
4. The entry with the latest date in field 10 ("Valid from") from the remaining entries is used. If there are also several entries concerning this criterion, the first wil be used.
5. 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, too, that means the desired price component cannot be determined.

3.3 Price factors

In the relational knowledge (table `Relation`), price factors can be indicated for a price item, which is bound to a variant condition. Therefore, the function *$SET_PRICING_FACTOR()* is used, which is specified in the OCD language sets as follows:

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

  The function defines the <Factor>, with which the price, fixed for the given <variant condition> (table `price.dat`), is supposed to be multiplied, if this <variant condition> is currently assigned to the variable `$VARCOND`.
  The call of the function usually follows the assignment of the variant condition to the variable `$VARCOND` within the same relational knowledge.
  <Factor> is an arithmetical expression.

  Example:

  If the value "Set 1" is assigned to the property "Electrification", the extra charge price is supposed to be deterimined in function of the width of the table. The price relation, which is bound to the relational object for the value "Set 2" 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 price factor can also be linked to a condition. The factor will be only used, if the condition is 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 the OCD:

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

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 which is not referenced in the scheme table is indicated, no special final article number is generated for the article. The final article number is then identical to the base article number.

The following shows a simple example:

> 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. The character string[25], indicated in the 3rd field of the scheme table, then follows. Finally, the separation character string is followed by the variant code, which is generated in the single 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 abbreviation "`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.

[25] This character string must not be empty. If field 3 is empty, a character string of one single blank character is used.

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> | @ | <Char>,] 1:n

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

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

<Char> := Every character except '`@`'

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. The formattings from the property table and the replacement and formatting instructions from the fields 6 to 10 of the scheme table are taken into account.
- *'@'* 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 in the current position.

In the scheme description, the example then is:
`Cupboard:Hight,_,@,@,-,@,@,_,Cupboard:Surface`
the final article number:
`5H_08-15_03`

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 par. 2.12).
- 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, it must not be identical to the separation character from field 4 of the scheme table for the predefined scheme `ValueList`.
- If field 10 is not empty and contains an even number of characters, the first half of the character string is placed first 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 paragraph 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 par. 2.9).

Principally, the text describing the property is made out of the (single line) designation of the property (from table `PropertyText`) and out of the text of the currently assigned value. Property value texts (table `PropValueText`) can be of several lines. 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 is always displayed there by its (single line) description and the first line of the text of the currently assigned value.

The following codes can be used:

0 The first line of the property description is made out of the property descpription and the first line of the text of the current property value. The remaining lines of the property value text follow.

For multivalued properties (see field *TxtControl* in the property table, par. 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 par. 2.12) and the indication from the field *MO_Sep* of the scheme table (par. 2.22)is used as separation character string between the values (resp. "**,**", if the field is empty).

This is a 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 description 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 designation only. The remaining lines correspond to the lines 2..n of the text of the current property value.

Example:
A cupboard has the property "Compartment bottoms reinforced" with the values "Yes" and "No".
The field *SuppresTxt* 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", "SuppresTxt" is set on False. The following property description appears
"Compartment bottoms reinforced"
instead of
"Compartment bottoms reinforced: 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 designation and first line of the property value text are suppressed.

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 *SuppresTxt* is set on True for all values of the property in the table `PropertyValue`.

It is senseful to use this mode for example for helping 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 "Colour cover" is released. If "Yes" is selected however, the properties "Colour seat cover" and "Colour back cover" are released.
The property "Two-colour cover" shall not/does not have to be described in the form in this case. This is done by code 4.

5 In particular valid for multivalued properties:
The first line of the property description consists of the property designation. The (possibly multi-line) designations of the currently set values follow according to the order which is determined by the position indication in the property value table (see par. 2.12). The designations of the first to the last value but one end with the character string indicated in the field *MO_Sep* of the scheme table (par. 2.22)(resp. ",", if the field is empty).

6 The determination of packaging data

This paragraph 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 (par. 2.5) and relational knowledge (par. 2.14 and 2.15).

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

1. Determination of the basic data of the article in the basic design by reading out the entry without variant condition (emply 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:
    Reading out of the corresponding entry from the table `Packaging` and addition of the amounts of all not empty fields to the corresponding amounts from the basic data resp. to the amounts possibly accumulated by previous variant conditions.

To step 2.1:

- Packaging relations are marked by the type of use 'PCKG' in the table `RelationObj` (field 4) and have to be of the type Action ('3') (field 3).
- Variant conditions are derived by assigning the variant condition to the special variable `$VARCOND` in the packaging relations (table `Relation`).
- The relevant packaging relations are determined according to the mentioned order from the relational objects
    1. of the article
    2. of the currently evaluated properties of the article
    3. of the current propery values

    .

A Language definition OCD_1

Preliminary remark:
This language definition corresponds to the language for relational knowledge from the format specification for `OCD 1.0`.

- This simple language allows to indicate conditions (all types of relations) and of assignments to the properties of an article (within actions only). Several assignments can take place within an action. They have to be separated by a comma.
- In logical and arithmetic expressions, property names can be used in the sense of variables. In the evaluation of the expression, they are replaced by the current value of the property.
- **Conditions**:
  - Conditions are simple or complex Boolean (logical) expressions. A logical expression is evaluated either as *true* or *false*. Complex Boolean expressions are built with the help of the operators `AND` and `OR` from subexpressions.
    In linked `AND` and `OR` operators, the `AND` links are evaluated first. The evaluation order can be controlled by using brackets. Compare for example `A and B or C` with `A and (B or C)`.
  - Simple logical expressions are:
    * Comparisons
    * Negation
    * Special conditions
  - Comparisons are noted with the help of the known comparison operators: < (or `LT`), <= (or `LE`), = (or `EQ`), <> (or `NE`), => (or `GE`) and > (or `GT`).
    The operands on both sides of the comparison have to be of the same type (only character string or numeric). An operand can be a constant. Character string constants have to be enclosed in single quotation marks. A numeric operand can be a complex arithmetic expression built with the help of the arithmetic basic operations. Furthermore, the functions mentioned in the annex F can be used in arithmetic expressions.
  - 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 quantity>)`
      This condition is true, if the current value of the property indicated in the left operand is comprised in the value quantity indicated in the right operand. The values in the value quantity have to be separated by commas.
  - If a logical expression refers to a property which the corresponding article does not possess, the expression cannot be evaluated. The only exception is the `SPECIFIED` expression, that can be used to avoid *undefined logical expressions*.
    The following rules apply for the linking operators `AND` and `OR`:
    * The result of an `OR` link 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` link 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 obviously false, that means it is not violated, if the logical expression is undefined.
    - A selection condition is violated, if it is not obviously true, that means it is also violated, if the logical expression is undefined.

- **Assignments** are done via the assignment operator `=`. The left operand is a property or the special variable `$VARCOND` (see par. 3).
  The operands on both sides of the assignment have to be of the same type.
  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 obviously fulfilled (not, if the logical expression is undefined).

- Lower case and capital letters are not differentiated when indicating property names and keywords. Example: `IF` is identical with `if`.

B Language definition OCD_2

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

- Character strings can be linked with the help of the operator +.
- Numeric values can be converted into a character string with the help of the function STRING().
- The Boolean constant FALSE 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 scope for the property has been restricted to exactly one value.
  - A comparison (including the IN condition) is only possible, if the value scope has been restricted to exactly one value (otherwise the result of the expression is undefined).
- A syntax and semantics, which are described more fully below, are defined for the relation type *Constraint*.
- In constraints and actions, value derivations can be realized with the help of value combination tables and the function TABLE().
  The respective syntax and semantics are described more fully below.

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 derive value properties or to restrict value scopes.
- Constraints have always to be linked to articles (see par. 2.14). Thus, expressions about several properties of property classes of the articles can be made in one constraint.
- A constraint consists of up to four parts, each introduced by a keyword plus colon and ended by a point:
- **Objects:**

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

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

  - 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 parts: <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 part.

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

  This constraint part 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 part (see below) the relations simply represent conditions. If one of the relations is not fulfilled or no obvious expression 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 part, the relations can simoultaneously effect value derivations or value scope restrictions. 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 quantity examination*: <Property name> IN (<Value quantity>)

    The value of the property indicated on the left side must be included in the value quantity indicated in the right part 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 paragraph.

- **Inferences:**

  determines the properties for which values are supposed to be derived via the constraint or for which their value scope is supposed to be restricted[26].

  This constraint part is optional. It is omitted, if no values are to be derived or no value scopes to be restricted.

  The derivation or restriction is done with the help of the relations in the Restrictions part.

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

– *Value derivation 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[27].

Example:

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

– *Value scope restriction via* `IN` *construct*:

If the property is listed on the left side of the `IN` expression under `Inferences` and restrictable, the expression effects a restriction of value quantity of the property concerning the quantity indicated in the right part of the expression[28]. The quantity indicated in the right part 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 derivation and value scope restriction 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 paragraph.

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

[27] The comparison is thus always fulfilled.
[28] The `IN` relation is thus always fulfilled.

B.2 Table call

Value combination tables (see par. 2.19) can be accessed in actions and contraints with the help of the function `TABLE()`. The general syntax of the call is:

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

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

Actual parameters can be:

- numeric or character string constant
- Property variable

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

The semantics of the table call differs in actions and constraints.

B.2.1 Table call in actions

Within actions, value combination tables are used to derive values for properties.

The properties, for which values are to be derived via a table call, have to be indicated in the parameter list as actual parameter and to be equipped with the prefix "`$SELF.`". All other actual parameters serve as access key.
If no or all actual parameters are equipped with the prefix "`$SELF.`", values are derived for all those properties which are indicated as actual parameter, but are not evaluated at the time of the call.

There is no table access and thus no value derivation, if either no actual parameters could be defined as access key according to the above described determinations or if one of the actual parameters is not evaluated.

The table call has to lead to an obvious result for each property to be derived. Otherwise the call fails and there will be no value derivation.

B.2.2 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` part:

- If the `Inferences` part is missing or all actual parameters are evaluated, the table call serves solely to check the consistency:
  All actual parameters then serve as key to access the table. If one of the actual parameters is not evaluated in the current configuration (missing `Inferences` part), there will be no table access and the constraint is not fulfilled. If the table includes no entry for the key defined through the actual parameters, the constraint is not fulfilled either.
- If one or several properties, which were given over as actual parameters when calling the table, are not evaluated, the constraint serves to derive values for these properties. (The other actual parameters then serve as key.) If at least one of the properties, for which a

derivation shall take place, is not listed under `Inferences`, there will be no table access and the constraint is not fulfilled.

If the table access provides several possible values for one property, a corresponding restriction of the value scope takes place, if it is a restrictable property.

For a non restrictable property, the table access can provide one value only, which is then assigned to the property. Otherwise, the constraint is not fulfilled.

- If all actual parameters are property variables and if all these properties are restrictable, but not evaluated in the current configuration, there is no existing key for a table access.

  In this case, the table call is handled as follows:
  All characters of the table are read out one after the other. Each property from the actual parameter list is checked on whether the value from the table is included in the currently restricted value quantity. If the values of all concerned properties of the table line are valid, the values are taken over each into a list per property (a multiple appearance of values is excluded). After the processing of all table lines, the such found value quantities are assigned to the respective properties as new (restricted) value quantity.

C Language definition OCD_3

This language definition comprises all determinations from the language definition OCD_2. Furthermore, the following further determinations for the processing of multivalued properties and for the 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 only.

  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 quantities of the set values 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 par. 2.6), it is necessary to differentiate, whether a property of the article currently to configure or a property of a superordinate article is referenced. Therefore, the following classifiers for properties are defined:

- `$self`:
  refers to the currently configured article
- `$parent`:
  refers to the directly 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 SAP_3_1

This language definition corresponds, except for the below mentioned exceptions, to the definition of the language which is used in the SAP R/3 system, release 3.1, to code relational knowledge. It is therefore referred to the corresponding SAP documentation.

Syntactically, there are the following restrictions:

- The builtin conditions `PART_OF` and `SUBPART_OF` are not supported.
- The builtin functions `SUM_PARTS`, `COUNT_PARTS`, `SET_DEFAULT` and `DEL_DEFAULT` are not supported.
- Function modules and their call via `FUNCTION` are not supported.

E Language definition SAP_4_6

This language definition corresponds, except for the exceptions mentioned in `SAP_3_1`, to the definition of the language which is used in the SAP R/3 system, release 4.6, to code relational knowledge. It is therefore referred to the corresponding SAP documentation.

F Arithmetic functions in relational knowledge

The following functions can be used in arithmetic expressions in the code of relations[29]. 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. An exception is initiated. If x is negative, y must be whole-numbered. If x is 0, y must be negative. 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 amount of x.

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

floor(x(Float)) → Float
: The function $floor()$ calculates the biggest whole-numbered 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 whole-numbered part of x.

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

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

G 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 an efficient execution 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), 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
  - Identification number clearly assigned in the context of the → EAN.UCC system for an item (product or service), which can be ordered and offset within business processes in trade.
  - Different defined code schemes can be used within the EAN.UCC system (EAN.UCC-14, EAN.UCC-13, EAN.UCC-8).
- **ILN**
  - Abbreviation for International Location Number
  - Identification number clearly 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.
  - There are 2 code schemes (type 1 and type 2), both 13-digit.
- **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 applied to transact business 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.

H Modification history

H.1 OCD 3.0 vs. OCD 2.1

- New levels "TX1" and "TX2" for tax rates in the price table (par. 2.16).
- New field *ScaleQuantity* in the price table (par. 2.16) to indicate *scale prices*.
- Support of *multivalued* properties.
  For this, new field *MultiOption* in property table (par. 2.9), new fields *MO_Sep* and *MO_Bracket* in the code scheme table (par. 2.22), as well as new enhanced language definition `OCD_3` (par. C) and removal of the restriction in the language definitions `SAP_3_1` and `SAP_4_6`.
- New concept of the *composite articles* (replaces the old set concept):
  - New article type "CS" (instead of "S") in the article table (par. 2.2).
  - New table `Composite` (par. 2.6).
  - New table `BillOfItems` (par. 2.7) replaces old table `Set`.
  - New area of use "BOI" for relations (par. 2.14).
  - Enhanced language definition `OCD_3` (par. C) and removal of the restriction on the language definitions `SAP_3_1` and `SAP_4_6`.
- New optional tables `Series` (par. 2.17) and `SeriesText` (par. 2.18).
- New optional tables `ArticleIdentification` (Abschn. 2.3), `PropertyIdentification` (Abschn. 2.10) and `PropValueIdentification` (Abschn. 2.13), as well as generalization (modification) of the identification table (par. 2.20).
- Diverse precisions and concretisations.

H.2 OCD 2.1 vs. OCD 2.0

- New pre-defined identification types `ILN-1` and `ILN-2` in the identification table (par. 2.20).
- New field *TxtControl* in the property table (par. 2.9) to control the property text in commercial forms.
- Property value texts can now be of several lines.
- New paragraph to describe the property text control.
- New field *Comment* in the version information table (par. 2.21).
- Concretisations concerning the language sets.

H.3 OCD 2.0 vs. OCD 1.0

- Length dimensions/restriction of fields of the data type `Char` only in well-founded cases.
- New field *OrderUnit* in the article table (par. 2.2).
- New field *VariantCode* in the table `Identification` (par. 2.20).
- New tables `ClassificationData` (par. 2.4) and `ClassificationText` (par. 2.18).

- New field *RelObjID* in the property class table (par. 2.8).
- New scope for properties "RG" (par. 2.9).
- New field *Restrictable* in the property table (par. 2.9).
- New field *SuppressTxt* in the property value table (par. 2.12).
- Now several interval values for a property possible, if those are equipped with preconditions each excluding each other (par. 2.12).
- New relation type *Constraint* and a more exact description of the individual relation types (par. 2.14).
- New field *FixValue* in the price table (par. 2.16), but no percentage sign in the currency field allowed any more.
- Alternative table definition for text tables to indicate manufacturer and trade specific texts (par. 2.18.
- Pre-defined final article number schemes can also be article specifically parametrized now (paragraphs 2.22 and 4).
- New field *RelCoding* in the version information table (par. 2.21).
- Separate paragraphs in the annex to describe the different possible languages to code relational knowledge.