Additional Information
SAP EHS Management as Part of SAP ERP
Document Version: 1.4 – 2021-06-22

CUSTOMER

Data Import with the EH&S Open Content Connector

Typographic Conventions

Type Style	Description
Example	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons, labels, menu names, menu paths and menu options. Textual cross-references to other documents.
Example	Emphasized words or expressions.
`EXAMPLE`	Technical names of system objects. These include report names, program names, transaction codes, table names and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
`Example`	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text and names of installation, upgrade and database tools.
`Example`	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE`	Keys on the keyboard, for example, `F2` or `ENTER`.

Document History

Version	Date	Change
1.0	2014-12-15	Initial version
1.1	2015-02-20	Data format DATE added
1.2	2015-07-07	New dialog elements for logging and default data source
1.3	2020-01-21	New usage of language-dependent identifiers
1.4	2021-06-22	New note in section 2.2.2.7.8 User-Defined Texts (F:) added

Table of Contents

List of Figures

List of Tables

1 Introduction

You can use the *EH&S Open Content Connector (EH&S OCC)* to import phrases and data for specifications from formatted table-based files into the specification database.

You can import data with or without a system connection. When you import data with a system connection (online), the *EH&S OCC* loads the data directly into the specification database. When you import data without a system connection (offline), the program converts the data into a file that has the standard import format for specification data. You then use the *Import Specification* transaction (`CG33`) to import this transfer file into *Specification Management* (EHS-BD-SPE).

The phrase import is an enhancement to the specification import. The *EH&S OCC* converts data from a table-based file into an import file that has the standard import format for phrases. You then use the *Import Phrases* transaction (`CG31`) to import this transfer file into *Phrase Management* (EHS-BD-PHR).

The data import function of the *EH&S OCC* allows you to load data from external systems into the specification database more easily. This simplifies the process of migrating data from your legacy system, for example.

2 Preparing Data for Specification Import

2.1 Structure of the Specification Data Import File

To load specification data into the specification database, you must provide data in a table-based format, such as a relational database, Microsoft Excel file or a text file.

In most cases, data is exported from an external software system or database into a table-based file. This import file can have the following format:

SubID	CAS Number	Boiling Point	...
A	67-56-1	65 °C	...
B	110-80-5	78 °C	...
C	60-29-7	36 °C	...
...	...	...	...

Table 1: Structure of a specification data import file

The table contains a unique identifier of a specification (SubID) and data for characteristics. Multiple specifications and multiple characteristic instances can be included in a single table.

To use the powerful format conversion and data mapping features of the *EH&S OCC* specification import function, the data must be provided with a specific format. This is described in detail in the following sections.

In the import file, the first column of the table has a specific significance: It is always interpreted as a unique specification key (such as a SubID or any other unique identifier).

Data provided in one row of the import file can only belong to one specification. Data of different specifications must not be combined in a single row.

As the specification import function parses the table from top to bottom, the system saves the data for one specification when the value of the first column changes. The field can be left empty in subsequent rows for the same specification.

It does not matter which data field in the specification database is used as the specification key in the first column. SAP recommends you use the specification key or a unique identifier, such as the CAS number or EU number. It is not necessary to use a specification database field as key. The key can be an artificially-generated unique number which is not written into the specification database.

If the specification key is not specified in the first or in any other column, the system automatically generates an artificial key during import.

> **Note**
>
> The *EH&S OCC* needs a TXT file format to import data. For this reason, you need to convert all other file formats to pure text format with the table columns separated by tabulators before starting data import.

2.2 Header Rows

Add header rows to your table to describe the data format of the data provided in the import file.

When using the TXT data format, the header rows must contain the following information:

SubID	CAS-No	Density	...
H:SUBID	I:IDENT,IDTYPE="NUM", IDCAT="CAS"	M:1013_005_VALUE;1013_005	...
STRING	STRING	NUMWITHUNIT	...
ABC	12345	5 g/cm³	...
DEF	67890	7 kg/m³	...
DEF	10815	8 kg/m³	...
...	...	...	...

1. Column name
2. Property or identifier
3. Data type / field structure

Table 2: Example structure of import data

The **first header row** contains the name of the column, the **second header row** maps the data in the import file to fields in the specification database. The mapping syntax follows the fact mapping of EH&S Expert, but also uses some enhancements.

For further information, see *EHS Expert* documentation in the Help Portal under http://help.sap.com/ehs27. Choose your release, go to the *SAP Library* section. Open the application help documentation and go to *Basic Data and Tools (EHS-BD) -> Specification Management (EHS-BD-SPE) -> Specification Editing -> Secondary Data Determination -> EH&S Expert.*

The **third header row** describes the data structure and data type of the data in each column. There are keywords for the usual data types that can be combined to describe more complex structures. In addition, regular expressions can be used to describe even the most sophisticated and fuzzy data contents. The column can be left empty for simple string or text data contents.

> **Note**
>
> Enter data that wish to import first in the fourth row of your table file.

2.2.1 First Header Row: Column Name

The first row describes the content in the columns and can contain any text. Subsequent columns may reference a preceding column by this name in one of the supported functions in the second or third header row. Cross-referencing columns is described in more detail in the following chapters..

2.2.2 Second Header Row: Mapping to Specification Database Fields

The second header row defines the mapping of the data in the import file to data fields in SAP *EHS Management*. A cell in the mapping row contains a description of one or more target fields in the specification database. The mapping syntax follows the *EH&S Expert* mapping syntax with slight modifications and enhancements.

The mapping information consists of different parts, for example a letter that defines the field in the specification database.
The most important first letters are included in the following table:

H:	ESTRH (specification header)
I:	ESTRI (identifier)
M:	Property in property tree
U:	ESTDU (usage)
F	ESTDF (user defined text)

Table 3: Examples for most important first letters

After a colon (:) one or more names of specification database fields follow. Multiple values have to be separated by a comma (,). Fields can also be provided with default values using the equal sign (=value). If data is to be imported into a characteristic, the characteristic ID und the property ID have to be separated by a semicolon (;).

Information that is provided after the colon is referred to as a "data element". A data element specifies the field in the specification database into which the data should be imported. It is part of the mapping information.

2.2.2.1 Importing Specification IDs

The first column usually contains the specification ID. In *Specification Management*, this ID is stored in table ESTRH in the field SUBID. Mapping information is as follows: H:SUBID.

> **Note**
>
> If you wish the system to generate a unique number as the specification ID during import, do not provide a specification ID in the import file.

2.2.2.2 Importing Identifiers

The sequence of the columns is not predefined. However, SAP recommends you enter the specification ID in the first column, and that you use the next to import identifiers to the specification.

In *Specification Management*, identifiers are stored in table ERSTI. Therefore, the first letter for mapping identifiers is "I".

The value "IDENT" follows a colon for the field in which the identifier is stored. Information about identifier type (field IDTYPE) and category (field IDCAT) follow, separated by commas.

If you wish to import a CAS number, mapping information could be as follows:

SubID	CAS Number
H:SUBID	**I:IDENT, IDTYPE="NUM", IDCAT="CAS"**
12108	50-00-0
25385	50-21-5
....	.....

Table 4: Mapping for importing one identifier

The CAS number is contained in the second column. In *Specification Management*, the CAS number is stored as an identifier of type NUM and category CAS. The values for IDTYPE and IDCAT can be initialized here with the fixed values "NUM" and "CAS". These two values must be separated by a comma.
For example: I:IDENT,IDTYPE="NUM",IDCAT="CAS"

If an identifier is language-dependent, you have to provide the language key. If you wish to import the identifier in only one language, you can initialize the field LANGU with a fixed value, such as DE for German, EN for English, or FR for French.

If you want to import a product name, for example, in German, mapping information could be as follows:

SubID	Product Name
H:SUBID	**I:IDENT, IDTYPE="NUM", IDCAT="CAS", LANGU="DE""**
12108	Formaldehyd
25385	Milchsäure
....	....

Table 5: Mapping for importing language-dependent identifier in one language

If you wish to import a product name, for example, in German, English, and French, you can provide the language key in a separate column. Mapping information could be as follows:

SubID	Language	Product Name
H:SUBID	I:LANGU	I:IDENT,IDTYPE="NAM", IDCAT="PROD"
12108	DE	Formaldehyd
	EN	Formaldehyde
	FR	formaldéhyde; aldéhyde formique
25385	DE	Milchsäure
	EN	lactic acid
	....	....

Table 6: Mapping for importing language-dependent identifiers in multiple languages

This example also shows that you can have more than one row for a specification. In this example, the product name is provided in different languages. In such cases, you can leave the column "SubID" empty until you wish to import data for the next specification. The specification import groups all data together in rows without a SubID automatically.

> **Note**
>
> For identifiers, the IDENT field must be the last column in the sequence of columns for one identifier. If there are other fields (such as LANGU) without an IDENT column to follow, the system writes an error in a log file; this data is not imported.

If you wish to import more than one identifier, you have to provide the data for the next identifier in the subsequent columns:

SubID	Language	Product Name	CAS Number
H:SUBID	I:LANGU	I:IDENT,IDTYPE="NAM", IDCAT="PROD"	I:IDENT,IDTYPE="NUM", IDCAT="CAS"
12108	DE	Formaldehyd	50-00-0
	EN	Formaldehyde	
	FR	formaldéhyde; aldéhyde formique	
25385	DE	Milchsäure	50-21-5
	EN	lactic acid	
	....	....	....

Table 7: Mapping for importing multiple identifiers

2.2.2.3 Importing Characteristics

You can import data that is stored in characteristics in *Specification Management*. For the mapping, you have to provide the characteristic ID and the property ID separated by a semicolon.

If you wish to import, for example, density data, mapping information could be as follows:

SubID	CAS No	Density Value	Density Temperature
H:SUBID	I:IDENT, IDTYPE="NUM", IDCAT="CAS"	M:SAP_EHS_1013_005_VALUE; SAP_EHS_1013_005	M: SAP_EHS_1013_005_EC_TEMP
		NUMWITHUNIT	NUMWITHUNIT
12346	67-56-1	0,79 g/cm³	20 °C
12345	60-29-7	0,71 g/cm³	20 °C
....	....	....	....

Table 8: Mapping for importing data to different characteristics in one property

In *Specification Management*, data for density is stored in characteristic SAP_EHS_1013_005_VALUE that is contained in property SAP_EHS_1013_005. The characteristic ID und the property ID have to be separated by a semicolon. The temperature, at which the density is measured, is stored in characteristic SAP_EHS_1013_005_EC_TEMP. Since this characteristic is also contained in property SAP_EHS_1013_005, you do not need to repeat the property information.

If you have density values measured at different temperatures, you have to enter a separate row for the subsequent density values:

SubID	CAS No	Density Value	Density Temperature
H:SUBID	I:IDENT, IDTYPE="NUM", IDCAT="CAS"	M:SAP_EHS_1013_005_VALUE; SAP_EHS_1013_005	M: SAP_EHS_1013_005_EC_TEMP
		NUMWITHUNIT	NUMWITHUNIT
12345	67-56-1	0,79 g/cm³	20 °C
		0,80 g/cm³	10 °C
12346	60-29-7	0,71 g/cm³	20 °C
....	....	....	....

Table 9: Mapping for importing data to different instances of a characteristic

If you provide the different density values for one specification as described in the table, the data is imported into two instances in the specification database.

> **Note**
>
> The third header row defines the data type and is discussed below. Here, all columns are of the type “string”, except for the density value in the last column. This is defined as a “number” for a numerical value.

In *Specification Management*, characteristics can be assigned multiple values. This is usually the case for characteristics that have phrases assigned. If you wish to import multiple values to one characteristic, you have to enter the different values in one row, separated by commas:

SubID	CAS No	Extinguishing Media - Suitable
H:SUBID	**I:IDENT, IDTYPE=“NUM“, IDCAT=“CAS“**	**M: SAP_EHS_1016_001_YES; SAP_EHS_1016_001**
		STRING.LIST<“,“>
12345	67-56-1	CUST-N05.00116610,CUST-N05.00116620, CUST-N05.00116570
12346	60-29-7	CUST-N05.00116590, CUST-N05.00118030
....	....	....

Table 10: Mapping for importing multiple values to one instance of a characteristic

Data for suitable extinguishing media is stored in characteristic SAP_EHS_1016_001_YES that is contained in property SAP_EHS_1016_001. The phrases that are separated by comma are imported to the characteristic *Suitable* together. After the import, specification 12345 will contain 3 suitable extinguishing media and specification 12346 will contain two suitable extinguishing media.

> **Note**
>
> You can import text instead of a phrase key if the text is available as a phrase in the specification database. The import function compares the text in the import file with the text of the available phrases. If a match can be determined, the phrase key of this phrase will be imported into the characteristic. For more information about phrase import, see the corresponding section in this document.

If you wish to import data to more than one characteristic, you have to provide the data for the next characteristic in subsequent columns:

SubID	CAS No	Density Value	Molar Mass
H:SUBID	I:IDENT, IDTYPE="NUM", IDCAT="CAS"	M:SAP_EHS_1013_005_VALUE; SAP_EHS_1013_005	M: SAP_EHS_1013_037_VALUE; SAP_EHS_1013_037
		NUMWITHUNIT	NUMWITHUNIT
12345	67-56-1	0,79 g/cm³	32,04 g /mol
....	....	....	....

Table 11: Mapping for importing data to different properties

SAP recommends you provide all data for one property in subsequent columns before starting with data for another property:

Density Value	Density Temperature	pH Value	pH Temperature	Molar Mass	....
....	....	....	....	....	....

Table 12: Example of a sequence of columns for importing data to multiple properties

Further examples are provided in the appendix.

2.2.2.4 Syntax

```
command              ::= { $T | $C }.                    (to control type definitions, etc. see below)
mapping              ::= <data element list>[;<char cat>].
data elem list       ::= < data element>[,< data element list>].
data element         ::= [[][<type>]:<data elem id>[=<init>][,<alternatives>]].
type   ::= letter.                                  (M=property,U=usage,I=identifier,... see EHS-Expert)
data elem id         ::= data element.                          (e.g. SAP_EHS_1013_005_VALUE)
alternative          ::= <data elem id>[=<Init>][,<alternative>].          (target alternatives)
init   ::= { <number> | "<string>" }.                                  (to initialize date values)
number ::= floating point numerical value.
string ::= alphanumeric text in quotes.
char cat             ::= name of characteristic category.
Legend:
  left ::= right              Key on the left side consists of expressions on the right side.
  X | A   Either X or A.
  { Expression }              Scope of OR expression.
  [Expression ]               Optional expression.
  .       End of expression. May be followed by a comment.
  <expression>                Expression is a key which is resolved further below.
  Text    Self-explaining text.
  Characters in bold and non-italics are control characters.
```

Figure 1: Syntax of target expressions

A target expression either defines a command ($...) or the mapping definition. The command T$ is described below together with type definitions. $C indicates that a data column is not to be imported. This may be useful if only parts of a data table should be imported at a particular time.

A mapping definition starts with a type letter, followed by a colon and a list of data elements. At the end of the list, the name of a characteristic category may be appended. You can also precede each data element in the list with the type letter.

2.2.2.5 Types of First Letters

This is the list of valid type letters:

Type Letter		Meaning	EHS Table	
X	*	“Dummy”, data is not mapped to an EHS data element	- - -	
1	*	Phrase catalog	TCG61	
2	*	Language-dependent name of phrase library catalog	TCG62	
3	*	Phrase group	TCG63	
4	*	Language-dependent name of phrase group	TCG64	
h	*	Phrase header	ESTPH	
p	*	Phrase key and phrase text	ESTPP	
o	*	Phrase origin	ESTPO	
j	*	Join of phrase to phrase set	ESTPJ	
H	*	Specification header	ESTRH	
D	*	Specification reference	ESTRR	
I	*	Identifier	ESTRI	
J	*	Material join	ESTMJ	
N	**	Like X but for instance-related data	- - -	
M	**	Property	Class system	
A	**	Instance administration data	ESTVA	
F	***	User-defined text	ESTDF	Additional information
U	***	Usage	ESTDU	
S	***	Literature source	ESTDS	
R	***	Assessment	ESTDR	
L	***	Composition	ESTVP	
7	***	Transport approval	EST07	
b	***	Packaging approval	EST0B	
d	***	Dangerous goods classification	EST0D	

Type Letter		Meaning	EHS Table	
f	***	Transport classification	ESTOF	
k	***	Packaging requirements	CCUNTPACK	
v	***	Special packaging provisions	CCUNTSPROV	
c	***	Provisions for transport	CCUNTCARR	

Table 13: Mapping types: Mapping data elements and additional data

Note

*) Targets X, 1, 2, 3, 4, h, p, o. j, H, D, I and J cannot include a characteristic category. They are ignored and a warning is issued in the log.

**)Targets of type N, A and M carry additional functionality. The left-most column of this type for a specific characteristic category triggers the creation of a new instance for the characteristic category. This is not carried out for additional information columns.

The first target of type N, A or M for a new characteristic category must be appended to the name of the characteristic category. In further targets for the same characteristic category, you may specify the name of the characteristic category, but this is not necessary.

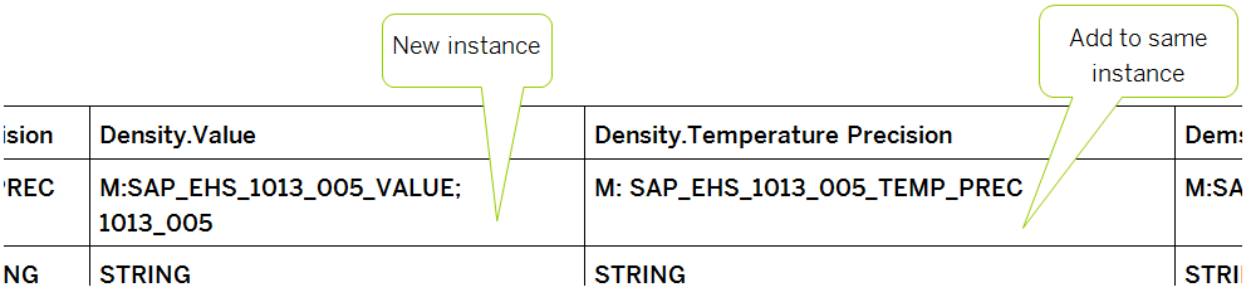


Table 14: Properties and instances

If targets of characteristic categories (cc) are mixed, then each first target for a characteristic category must include the name of the characteristic category again.

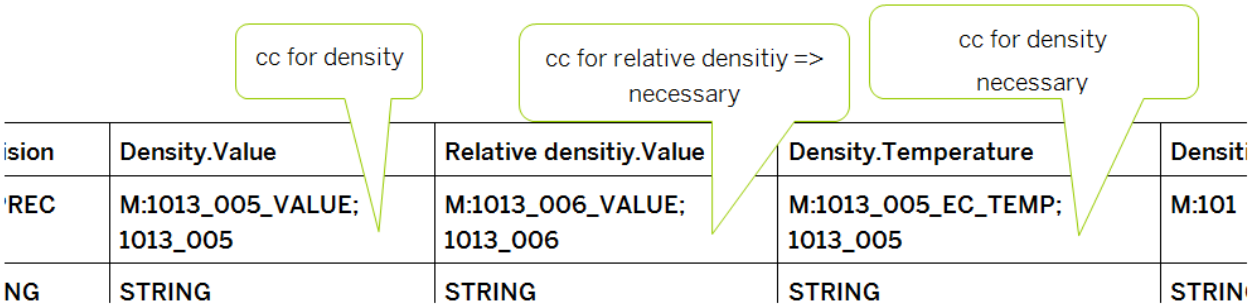


Table 15: Mixing characteristic categories

Caution

This is valid only for data fields when they are not empty. If a data field of a target of type N, A or M is empty, no new instance is created even though the name of a characteristic category is specified.

ision	Density.Value	Usage.Validity Area	Usage.Rating	Relative
'REC	M:1013_005_VALUE; 1013_005	U:VACLID	U:RVLID	M:1013_ 1013_00€
NG	STRING	STRING	STRING	STRING
	45 g/cm³	REG_WORLD	INTERNAL	
		DE	OFFICIAL	ca.
	80 g/cm³	REG_EU	PUBLIC	

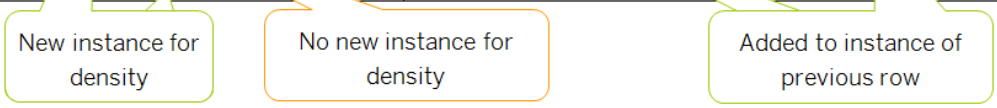
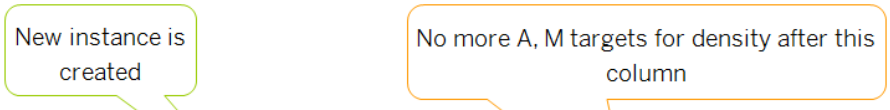


Table 16: Instances in multiple table rows

This makes it possible to add multiple and additional information to a single instance by simply adding it in additional rows. If no instance is previously created, an error message is issued.

The sequence of targets for a characteristic instance within a table row must follow the rule that targets of type N, M and A must occur before additional information targets. An error message is written into the log file otherwise.



ision	Density.Value	Density.Method	Usage:Validity Area	Usage:Rating
'REC	M:1013_005_VALUE; 1013_005	M:1013_005_METHOD	U:VALAREA	U:RVLID
NG	STRING	STRING	STRING	STRING

Table 17: Sequence of targets

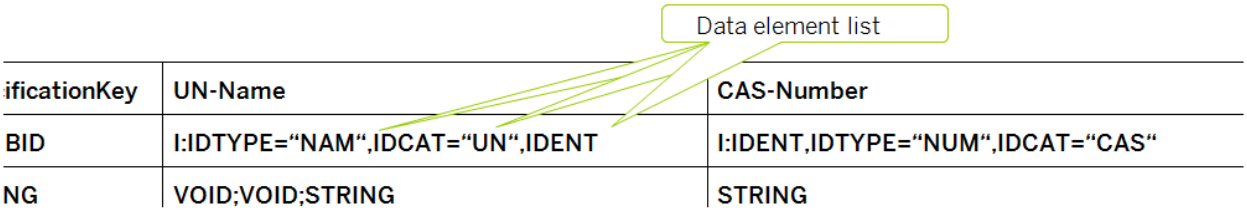
The rule does not apply for targets of type H, D, I and J, as these are independent of characteristic instances.

The sequence of targets of type N, A and M itself is irrelevant. These targets can be mixed in any order within the same characteristic category. The first one creates the instance.

> **Note**
>
> ***) Targets of type F, U, S, R, L, 7, b, d and f contain additional information for a characteristic category and must be placed behind targets of type A, M and N within the same characteristic category to avoid ambiguities.

Column contents can be mapped to more than one EHS data element by specifying a comma-separated list of data elements as the target. A data element in the list is either a property name (such as, SAP_EHS_1013_005_VALUE), or a table field (IDENT), depending on the target type. Optionally, data elements can be pre-initialized. If a data field of this column is empty, it is filled with this initial value. All data elements of a column must belong to the same characteristic category.

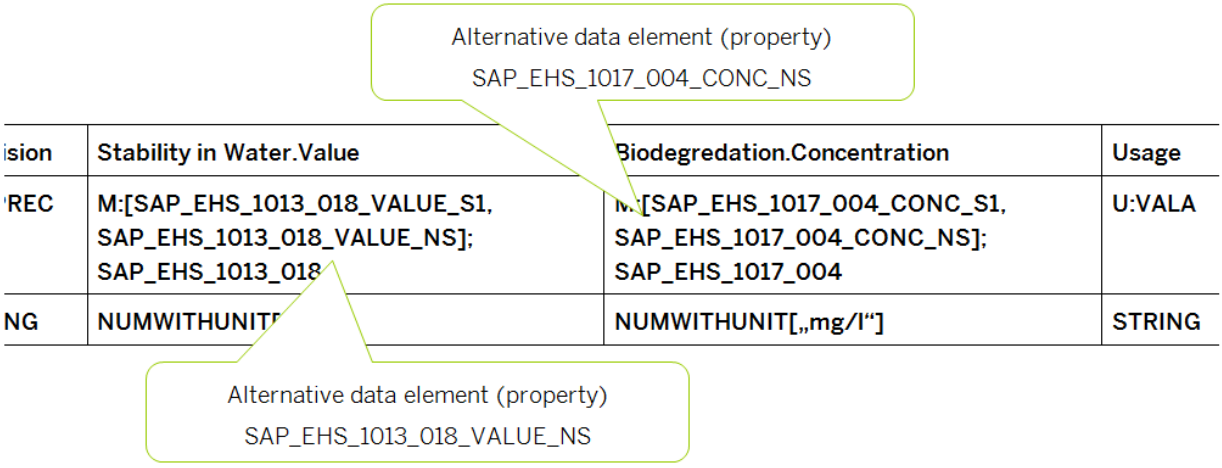
2.2.2.6 Data Elements



ificationKey	UN-Name	CAS-Number
BID	I:IDTYPE="NAM",IDCAT="UN",IDENT	I:IDENT,IDTYPE="NUM",IDCAT="CAS"
NG	VOID;VOID;STRING	STRING

Table 18: Lists of data elements and initialization

A group of data elements in brackets defines alternatives. The correct alternative for a specific value is determined by the type definition (see example below). A single data element in brackets is ignored and a warning is written into the error log.



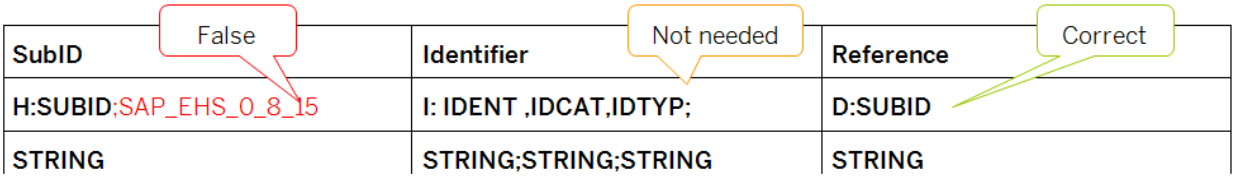
ision	Stability in Water.Value	Biodegredation.Concentration	Usage
REC	M:[SAP_EHS_1013_018_VALUE_S1, SAP_EHS_1013_018_VALUE_NS]; SAP_EHS_1013_018	M:[SAP_EHS_1017_004_CONC_S1, SAP_EHS_1017_004_CONC_NS]; SAP_EHS_1017_004	U:VALA
NG	NUMWITHUNIT	NUMWITHUNIT[„mg/l"]	STRING

Table 19: Alternative data elements

Alternative data elements can also be initialized to force a value if an alternative is not used by the data itself. Initialization values can be either numerical or a text in inverted commas.

2.2.2.7 Examples

2.2.2.7.1 Specification Header (H:)



SubID	Identifier	Reference
H:SUBID;SAP_EHS_0_8_15	I: IDENT ,IDCAT,IDTYP;	D:SUBID
STRING	STRING;STRING;STRING	STRING

Table 20: False/Right: H, D- and I target fields

For an H target field, a value assignment type (VAT) is not required. The semicolon in the identifier column is not required since no SBA is given which is not allowed for I target fields anyway.

2.2.2.7.2 Identifiers (I:)

Sublist	Identifier
I:SUBLIST	I:IDENT,IDTYPE="NAM",IDCAT="CAS",LANGU="DE"
STRING	STRING
ADNR	IdentifierXY

Table 21: I target fields

Identifier data can be separated in several columns. The column that contains the identifier name (table field IDENT) must then be the last column that relates to the identifier.

2.2.2.7.3 Usage (U:)

You can add a usage to your identifiers:

Identifier	Rating	Validity Area
I:IDENT,IDTYPE="NAM",IDCAT="PROD"	U:VACLID;I:NAM,PROD	U:RVLID;I:NAM,PROD
STRING	STRING	STRING
formaldehyd	PUBLIC	EN

Table 22: U target fields

2.2.2.7.4 Specification Reference (D:)

You can reference with a SubID or an identifier:

Reference Specification
D:SUBID
STRING
0000000001

Table 23: D target fields

Or:

Reference Specification
D:IDENT,NAM,PROD
STRING
Reference Spec 001

Table 24: Alternative for D target fields

2.2.2.7.5 Inheritance (E:)

Caution

You can only load an inheritance relationship in online mode. As there is no syntax for inheritance in the import files (.dat file extension), this does not work in offline mode.

If you wish to create an inheritance relationship, you must specify the inheritance-template, the template group and the target or source specification. The mapping character E is used for the source specification and the character V for the target.

Template Group	Template	Source
E:TEMPLATEGROUP	**E:TEMPLATE**	**E:SUBID**
STRING	**STRING**	**STRING**
TEMPLATE	ALL	0000000001

Table 25: E target fields

Or

Template Group	Template	Target
V:TEMPLATEGROUP	**V:TEMPLATE**	**V:SUBID**
STRING	**STRING**	**STRING**
TEMPLATE	ALL	0000000002

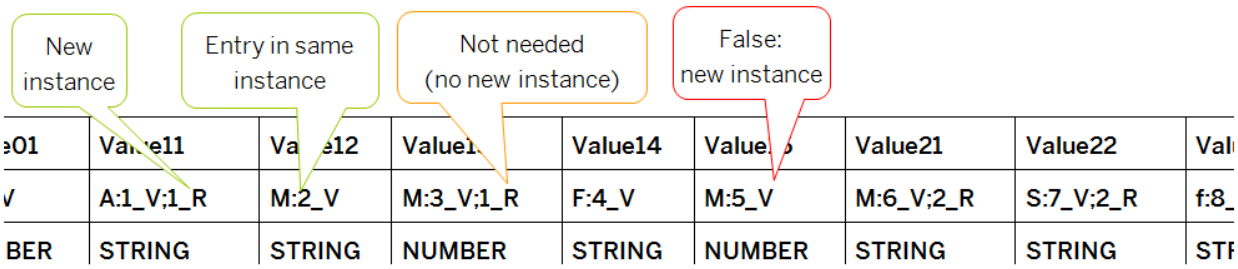
Table 26: V target fields

The specification import function only creates the inheritance relationship. The data inheritance itself from the source to the target specification is done by the standard inheritance job. Prior to ERP 6.0 EHP 3, this was an asynchronous job. It is therefore not possible to delete an existing inheritance relationship and create a new one in a single load. In this case, the job cannot be completed.

2.2.2.7.6 Material Assignment (J:)

The data type of the material number is a string. This is why you have to specify the exact material number, including leading zeroes. As an example, material number 0000072 is not the same as 72.

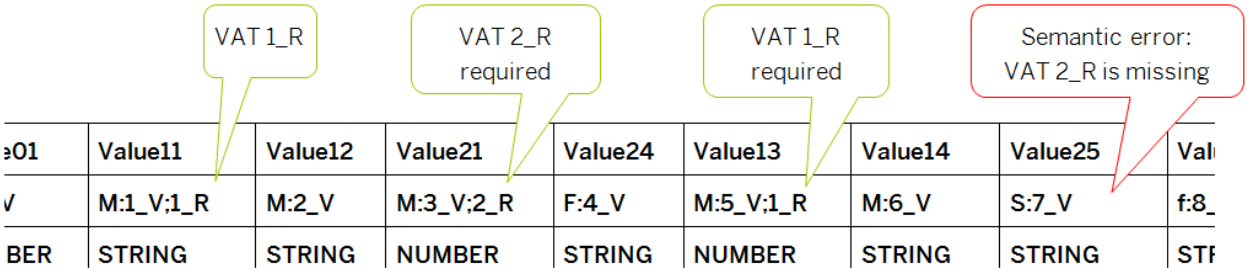
2.2.2.7.7 Characteristics (M: A:)



e01	Va e11	Va e12	Value1	Value14	Value	Value21	Value22	Val
V	A:1_V;1_R	M:2_V	M:3_V;1_R	F:4_V	M:5_V	M:6_V;2_R	S:7_V;2_R	f:8_
BER	STRING	STRING	NUMBER	STRING	NUMBER	STRING	STRING	STF

Table 27: False/Right: A- and M-target fields

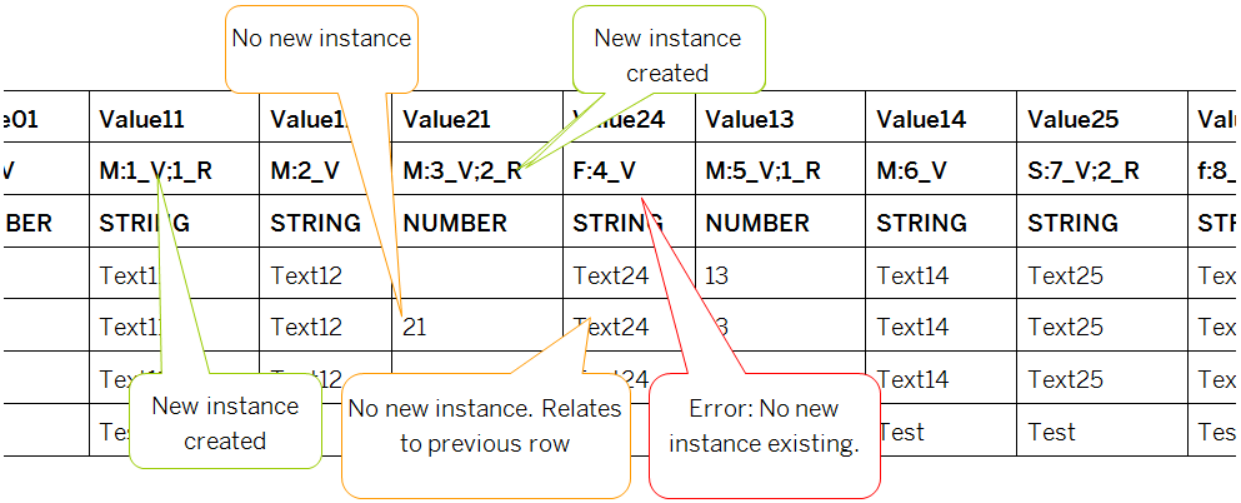
In column Value11, a new instance is created to which Value12 relates. Therefore, in this column no value assignment type is needed. In column Value13 the same value assignment type is still specified. This information is not required and will be ignored. As column Value14 specifies additional information (here a user-defined text), this column may not be followed by an M or A target. In column Value21, a new instance of value assignment type 2_R is created into which characteristic 6_V is written.



e01	Value11	Value12	Value21	Value24	Value13	Value14	Value25	Val
V	M:1_V;1_R	M:2_V	M:3_V;2_R	F:4_V	M:5_V;1_R	M:6_V	S:7_V	f:8_
BER	STRING	STRING	NUMBER	STRING	NUMBER	STRING	STRING	STF

Table 28: False/Right: Merging value assignment types

In column Value11, instance 1_R is created to which Value12 relates. In column Value21, an instance of VAT 2_R is created to which the next column relates. As in column Value13, a characteristic of VAT 1_R is targeted (Value1x stands for Values of VAT 1_R, Value2x for Values of VAT 2_R!), the VAT must be specified here. Column Value25 was meant to target the VAT 2_R. Because of column Value13 however, the current VAT is 1_R.

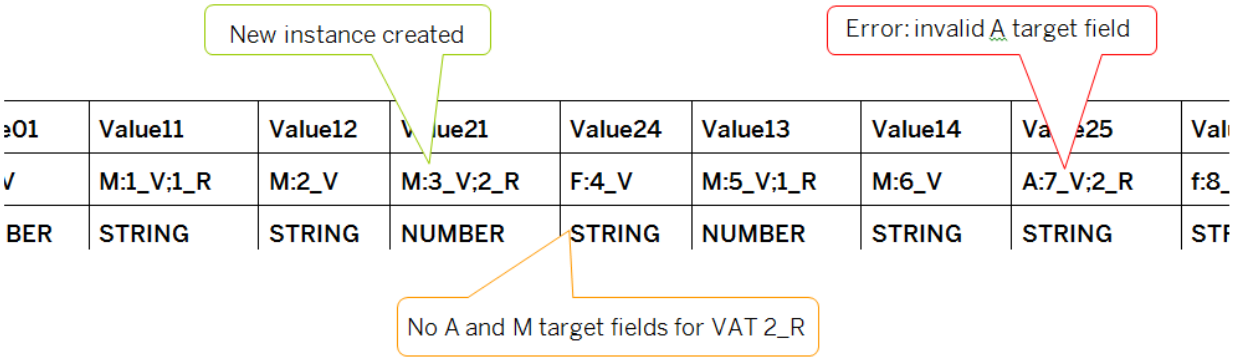


e01	Value11	Value1	Value21	ue24	Value13	Value14	Value25	Val
V	M:1_V;1_R	M:2_V	M:3_V;2_R	F:4_V	M:5_V;1_R	M:6_V	S:7_V;2_R	f:8_
BER	STRI G	STRING	NUMBER	STRIN	NUMBER	STRING	STRING	STF
	Text1	Text12		Text24	13	Text14	Text25	Tex
	Text1	Text12	21	Text24	3	Text14	Text25	Tex
	Tex	12		24		Text14	Text25	Tex
	Te					Test	Test	Tes

Table 29: False/Right: Creating instances

The content of the data fields are criteria for the creation of instances.

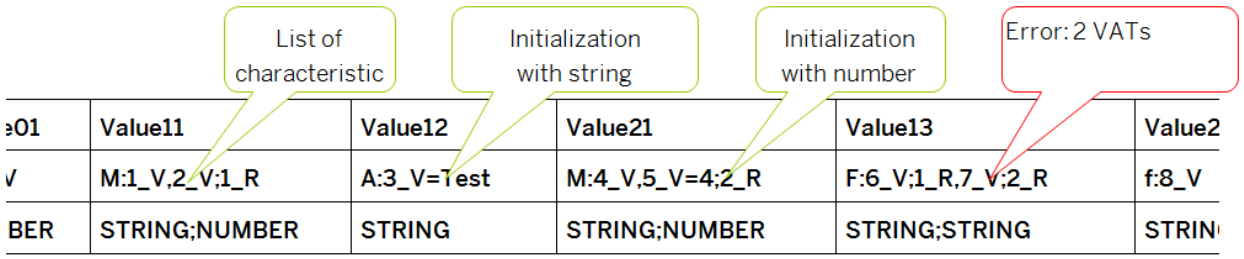
As there is no data in the 4th row of column Value21, a new instance of VAT 2_R will not yet be created in this row. The implicit reference to VAT 2_R in column Value24 is no error yet, since the entry relates to the last created instance even if that instance was created in the previous row. The problem is that this is the first data row and therefore no instance from a previous row exists.



e01	Value11	Value12	ue21	Value24	Value13	Value14	Va e25	Val
V	M:1_V;1_R	M:2_V	M:3_V;2_R	F:4_V	M:5_V;1_R	M:6_V	A:7_V;2_R	f:8_
BER	STRING	STRING	NUMBER	STRING	NUMBER	STRING	STRING	STF

Table 30: False/Right: Sorting target fields

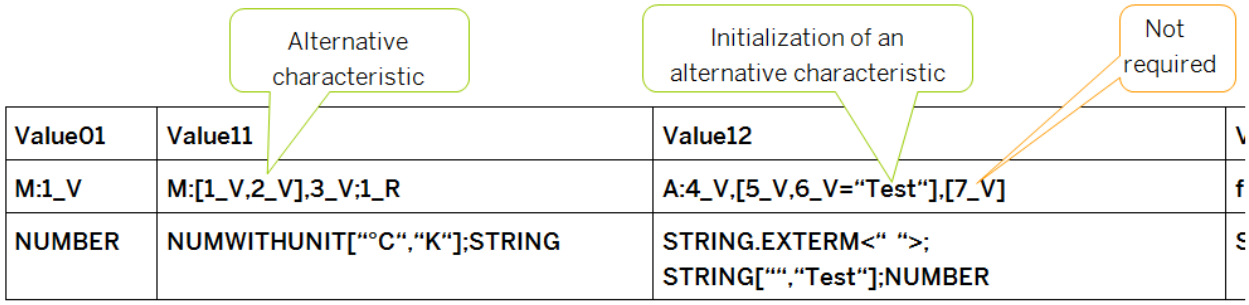
In column Value24, a user-defined text is written. The entry in column Value25 is invalid because after additional data no A- or M-Target field for the same VAT is permitted in the row.



e01	Value11	Value12	Value21	Value13	Value2
V	M:1_V,2_V;1_R	A:3_V=Test	M:4_V,5_V=4;2_R	F:6_V;1_R,7_V;2_R	f:8_V
BER	STRING;NUMBER	STRING	STRING;NUMBER	STRING;STRING	STRIN

Table 31: False/Right: Lists of characteristics and initializations

The data from column Value11 is written into 2 different characteristics of VAT 1_R. If there is no data in column Value12, the field is set to *Test* whenever it is empty.



Value01	Value11	Value12	V
M:1_V	M:[1_V,2_V],3_V;1_R	A:4_V,[5_V,6_V="Test"],[7_V]	f
NUMBER	NUMWITHUNIT["°C","K"];STRING	STRING.EXTERM<" ">; STRING["","Test"];NUMBER	S

Table 32: False/Right: Alternative characteristics

The data type NUMWITHUNIT writes its selection of units to the first characteristic. If a unit is found that does not correspond to the selection, the value is written to 2_V. In column Value12, the alternative characteristic 6_V is initialized with *Test*. If the alternative is not met, the characteristic 6_V contains *Test*. Brackets around single characteristics are ignored.

2.2.2.7.8 User-Defined Texts (F:)

A user-defined text is specified with the data element TEXT. This does not directly correspond to a field in the table ESTDF. The reason for this is that the table ESTDF can only handle text up to a certain length. Longer texts are saved in an additional long text table. The specification import function does this automatically. The column with the user-defined text needs to follow the last A- or M-Target field of the same VAT. (Graphic)

It is also possible to link to a DMS document. To do this, the document-specific fields of table ESTDF (DOKAR, DOKNR, DOKVR, DOKTL = Document type, Document number, Version, part of Document) have to be filled. In field TEXT, the given name of the document link must be specified. Example:

<Document name> (<DOKAR> <DOKNR> <DOKVR> <DOKTL>)

Note: For user defined texts, the TEXT field must be the last column in the sequence of columns for one user defined text. If there are other fields (such as LANGU) without a TEXT column to follow, the system writes an error in a log file; this data is not imported.

2.2.2.8 Examples

Density	Flash Point
M:SAP_EHS_1013_005_VALUE_PREC, [SAP_EHS_1013_005_VALUE,SAP_EHS_1013_005_ NS];SAP_EHS_1013_005	M:SAP_EHS_1014_009_VALUE_PREC="ca.", [SAP_EHS_1014_009_VALUE,SAP_EHS_1014_009_ NS];SAP_EHS_1014_009

(STRING.INTERM<“.“>); NUMWITHUNIT[“g/cm³“,“kg/m³“]	(STRING.INTERM<“.“>); NUMWITHUNIT[“°C“,“K“]

Table 33: Target examples: Alternatives, properties and characteristic categories

Here, data under the column *Density* is split into two properties: precision and value. The value is possible for one of two alternatives: value in standard unit and value in non-standard unit. The type definition in the third row defines that values with unit “g/cm3” and “kg/m3” shall be written into the first alternative SAP_EHS_1013_005_VALUE; all others into the second SAP_EHS_1013_005_VALUE_NS.

The precision is scanned up to a dot; “ca.” is potentially expected.

The flash point example is similar, but with different units to check. In addition, the precision is initialized to “ca.” so that every value is considered precisely.

SubID	CAS Number	Identifier2	Identifier3
H:SubID	I:IDENT, IDTYPE=“NUM“, IDCAT=“CAS“	I: IDENT ,IDTYPE,IDCAT	I: IDENT ,IDTYPE,IDCAT
STRING	STRING ; VOID; VOID	STRING.EXTERM<“ “>; STRING.EXTERM<“ “>; STRING	STRING.EXTERM<“ “>; STRING.EXTERM<“ “>; STRING

Table 34: Target example: Header, identifiers

The first column defines the specification ID. The subsequent columns contain various identifiers.

More examples follow in the chapter describing type definitions.

2.2.3 Specification References

References to other specifications in composition data, transport classification or the reference specification feature can be made by referencing to the specification key or a unique set of identifiers.

When importing the data, the specification import function checks whether the referenced specification exists and can be uniquely identified in the *SAP EHS Management* system. If it does not exist, it is created. If the identification is not unique, the import for this specific data item fails.

2.2.4 Third Header Row: Type Definitions and Functions

The third header row defines the data type for each column. The following data types are the most useful:

Type	Alias	Description
NUMBER	N	Numerical value
NUMWITHUNIT	NU	Numerical value with unit

Type	Alias	Description
PHRASE	P	Phrase (obsolete - is handled like STRING)
STRING	S	Any text (default)
VOID	V	Does not interpret data content, writes only predefined values.
DATE		Calendar Date

In addition to these simple types, useful functions can be appended to the type keyword with a period. There are functions to specify separator characters for concatenated values and to carry out standard string manipulation (extraction, conversion to upper case or lower case, etc.). The LIST function saves a concatenated value into a property with multiple values. Some functions allow cross references to data in other columns. Multiple types can be combined with semicolons to describe complex data structures inside one column to be targeted to multiple EHS data elements. The data content is then separated by blanks (default) or any other specified separator.

In addition to simple data types, data content can be described and separated by a regular expression following the POSIX standard.

2.2.4.1 Syntax

```
type definition      ::= <type list>.                                  (Usage of type definition)
type list            ::= <type expression>[;<type list>].               (Semicolon-separated list of types)
type expression      ::= [()<type>[<function list>][<selection list>][)].
type   ::= <string>.  (Type name see below.)
string ::= Alphanumerical text.
function list        ::= <function>[<function list>].
function             ::= <func> | {<reg exp parameter>}.
func   ::= .<function name><[<parameter list>]>.                       (see below)
function name        ::= <string>.                                     (see below)
parameter list       ::= <parameter>[,<parameter list>].               (depending on function)
parameter            ::= { <number> | "<string>" }.                    (string in quotes!)
number ::= floating point numerical value.
reg exp parameter    ::= <reg exp>[,<reg exp>[,<reg exp>]].            (search, replace, copy rest)
reg exp              ::= "<string>".                                   (regular expression in POSIX standard)
selection list       ::= <selection>[,<selection list>].               (depending on type)
selection            ::= [<list>].                                     (Definition of a selection set)
list   ::= <list element>[,<list>].
list element         ::= <string>.                                     (typabhängig)
Legend:
  left ::= right          Key on the left side consists of expressions on the right side.
  X | A   Either X or A.
  { Expression }          Scope of OR expression.
  [Expression ]           Optional expression.
  .       End of expression. May be followed by a comment.
  <expression>            Expression is a key which is resolved further below.
  Text    Self-explaining text.
  Characters in bold and non-italics are control characters.
```

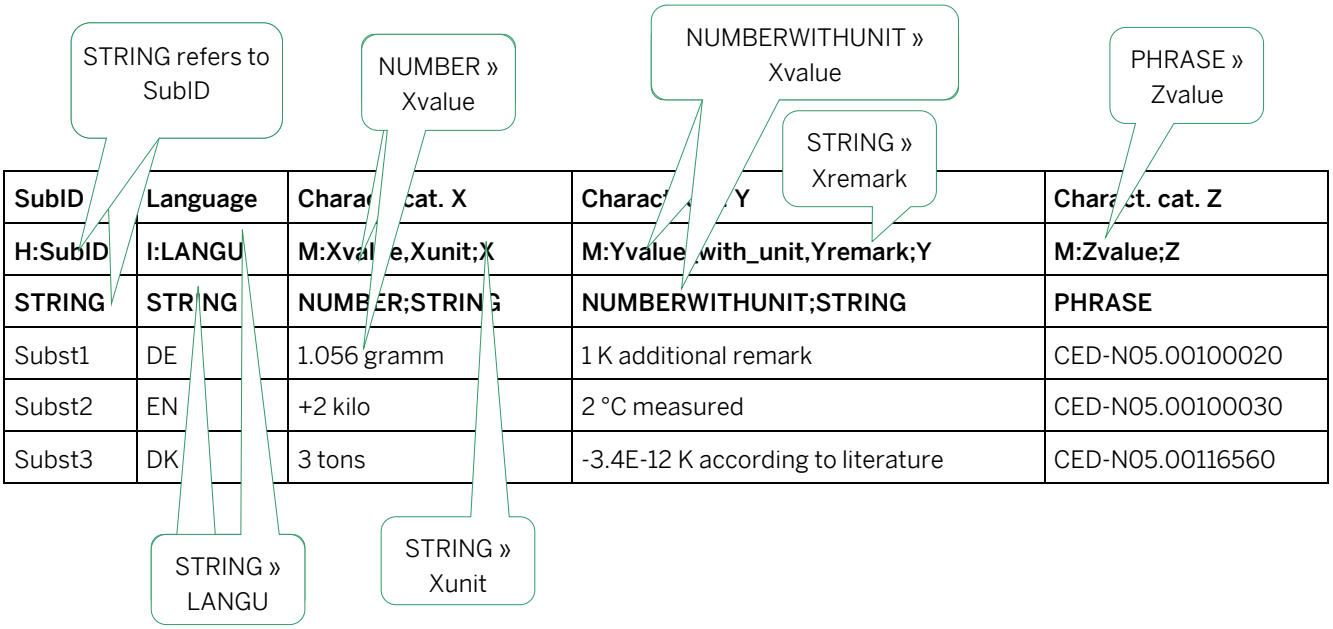
Figure 2: Syntax of type definitions

Type definitions can consist either of a single type expression or a list of type expressions, separated by semicolons. A type expression starts with a type name.

SubID	Language	Characteristic Cat. X	Characteristic Cat. Y	Characteristic Cat. Z
H:SubID	I:LANGU	M:Xvalue;X	M:Yvalue;Y	M:Zvalue;Z
STRING	STRING	NUMBER	NUMBERWITHUNIT	PHRASE

Table 35: Simple type definitions

Types can be combined if a column contains values for multiple data elements. The type expressions have to be specified in sequence with the data values and these are separated by semicolons.



SubID	Language	Charac cat. X	Charac Y	Char t. cat. Z
H:SubID	I:LANGU	M:Xval e,Xunit;X	M:Yvalue with_unit,Yremark;Y	M:Zvalue;Z
STRING	STR NG	NUMB ER;STRIN G	NUMBERWITHUNIT;STRING	PHRASE
Subst1	DE	1.056 gramm	1 K additional remark	CED-N05.00100020
Subst2	EN	+2 kilo	2 °C measured	CED-N05.00100030
Subst3	DK	3 tons	-3.4E-12 K according to literature	CED-N05.00116560

Table 36: List of type expressions

To split data values into parts to match the multiple types in a list, use whitespace characters to separate value parts. The NUMBER type obviously requires a numerical value, separated by a whitespace from the subsequent value part. NUMWITHUNIT looks for a numerical value and an additional word interpreted as unit. For data type STRING, a different separator character can be specified as whitespaces are often part of a string.

Types and value parts are referred to each other from left to right. If the number of types and value parts does not match, then any additional type or value part is ignored and a warning is written into the log file.

Optionally, predefined functions can be added to a type expression. Functions may need parameters, which may be strings in quotes or numerical values. For a complete list of functions and their parameters, see below.

SubID	Identifier	Density	Flash Point
H:SubID	**I:IDTYPE,IDCAT,IDENT**	**M:1013_005_VALUE; 1013_005**	**M:1014_009_VALUE_PREC, 1014_009_VALUE;1014_009**
STRING	**STRING.EXTERM<“,“>; STRING.EXTERM<“ “>; STRING**	**NUMWITHUNIT**	**(STRING.EXTERM<“ “>); NUMWITHUNIT**
ABC	NAM,OLD mineral water	5 g/cm³	300 °C
DEF	NUM,CAS 50-00-0	8 kg/m³	ca. 750 K

Table 37: Type lists and functions

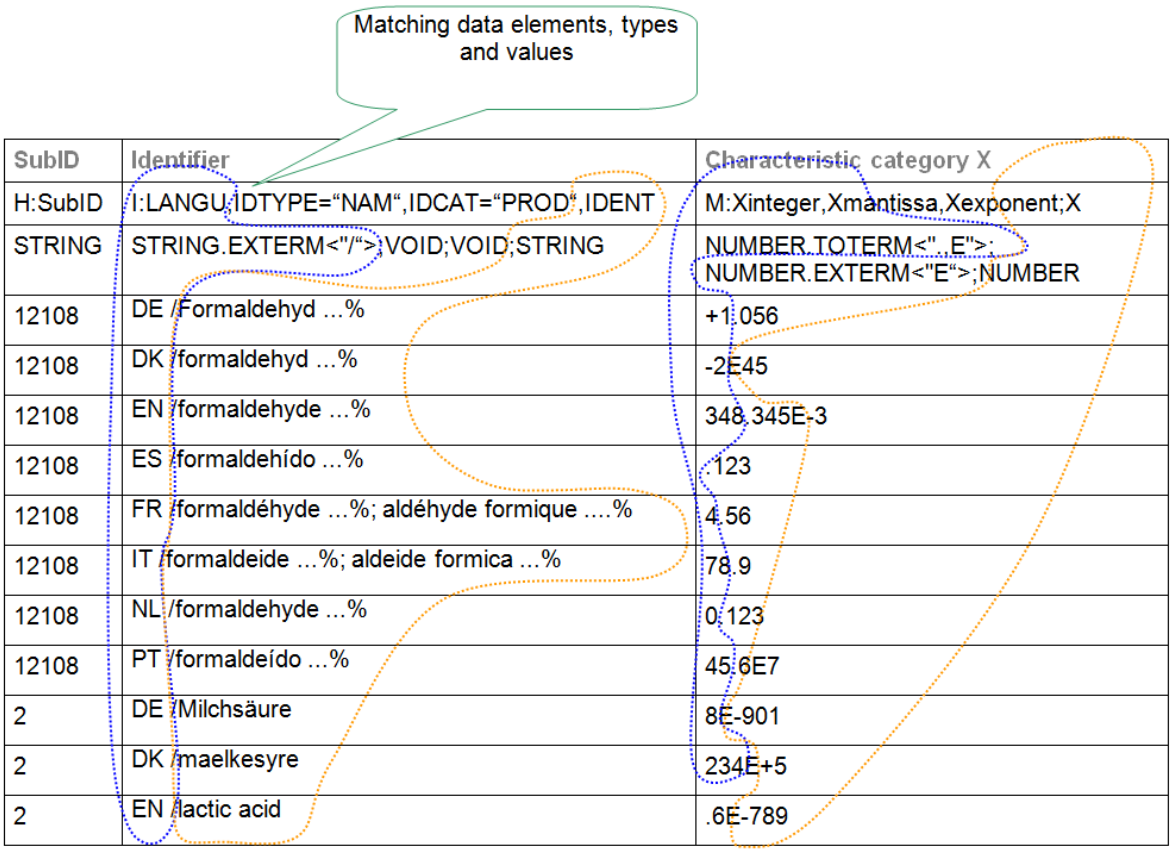
The values of the first column are defined as a single string.

The second column uses the function EXTERM to specify the separator character for dividing the value into three separate strings. The separator character is written as a parameter in angled brackets <>. The first string is parsed up to a comma and goes into IDTYPE. The function EXTERM excludes the separator character; therefore the first value

row delivers “NAM” as the value for IDTYPE. The second string continues after the comma up to and excluding the first blank character (EXTERM<” “>). Therefore *OLD* is put into IDCAT. The rest of the value (*mineral water*) is taken as the value for IDENT as there further functions are specified.

The third column shows the type NUMWITHUNIT. This type expects a numerical value followed by a unit string, here “5 g/cm3”. The unit string has to be separated by a blank character from the numerical value. Multiple whitespace characters are reduced to a single blank character.

The forth column puts the first type expression in parenthesis. This means that a match for the type expression is optional. The corresponding data element (here precision) is filled only if there is a match for the type expression, in this case in the second data row (“ca. 750 K”).



SubID	Identifier	Characteristic category X
H:SubID	I:LANGU;IDTYPE=“NAM“,IDCAT=“PROD“,IDENT	M:Xinteger,Xmantissa,Xexponent;X
STRING	STRING.EXTERM<"/">;VOID;VOID;STRING	NUMBER.TOTERM<",.E">; NUMBER.EXTERM<"E“>;NUMBER
12108	DE /Formaldehyd ...%	+1.056
12108	DK /formaldehyd ...%	-2E45
12108	EN /formaldehyde ...%	348.345E-3
12108	ES /formaldehído ...%	.123
12108	FR /formaldéhyde ...%; aldéhyde formique ....%	4.56
12108	IT /formaldeide ...%; aldeide formica ...%	78.9
12108	NL /formaldehyde ...%	0.123
12108	PT /formaldeído ...%	45.6E7
2	DE /Milchsäure	8E-901
2	DK /maelkesyre	234E+5
2	EN /lactic acid	.6E-789

Table 38: Example for functions

If value parts are not uniquely identifiable by fixed separator characters, regular expressions can be used to define text search patterns. Regular expressions are included in braces {} and quotes and are placed behind the type and function.

Types can be made optional by putting them in parenthesis. This is possible only at the beginning (one optional type only) and at the end (multiple optional types) of a type list.

A type expression can be completed by a selection list. Selection lists define the selection of alternatives from the data element mapping. A selection list is included in brackets [ ] and contains a list of possible data values separated by commas. If the actual data value matches one of the lists, it is written to the matching data element alternative. The sequence of selection lists and the sequence of alternatives must correspond. In this way, if the actual value is found in the first selection list, it is imported into the first alternative, if it is found in the second, it is imported into to the

second alternative, and so on. If it does not match any of the values in any selection list, it is imported into the last alternative. There should also be one more alternative to the selection lists. The type of values in a selection list depends on type and function.

SubID	Density	Condition	Flash point
H:SubID	M:[1013_005_VALUE, 1013_005_VALUE_NS];1013_005		M: 1014_009_VALUE_PREC, [1014_009_VALUE, 1014_009_VALUE_NS];1014_009
STRING	NUMWITHUNIT[“g/cm³“,“kg/m³“]		(STRING.EXTERM<“ “>), NUMWITHUNIT.CMPNC<2> [“Default“,“Standard“]
ABC	5 g/cm³	Standard	300 °C
DEF	8 kg/m³	Default	ca. 750 K
GHI	7 mg/mm³	No default	-10 Fahrenheit

Table 39: Example for selection lists

In the second column, there is a selection list for the type NUMWITHUNIT: [“g/cm³“,“kg/m³“]. Any value with one of these units is imported to the first alternative 1013_005_VALUE (data row ABC and DEF). Any value with any other unit is imported to the second and last alternative 1013_005_VALUE_NS (data row GHI). If there are not enough alternatives, the system issues a warning in the log file.

The fourth column solves the same problem with an additional column. The column *Condition* does not contain any mapping or type definition, but simply values for comparison. The CMPNC<2> function in the type definition of the third column references these values. 2 is the internal column number, column numbers start with 0. The selection list in the fourth column now checks for the helper values of the third column to decide which alternative to put the value into. Non-matching values (“No default”) are entered into the last alternative.

2.2.4.2 List of Data Types

Type	Alias	Description
NUMBER	NR	Numerical value
NUMWITHUNIT	NU	Numerical value with unit
PHRASE	PH	Phrase (obsolete - is handled like STRING)
STRING	S	Any text. This is the default and can be discarded.
TYPEDEF	TD	Special handling of types and data fields.
VOID	V	Ignore value, use initialization
DATE		Calendar Date

Table 40: List of data types

Aliases can optionally be used rather than the full type names. This is provided by the specification import macro functionality which is described below.

2.2.4.3 Macros

The specification import function reads an external configuration file which can define default options for the BAPI import and default options for the parsing process.

In addition, the user can define macros as abbreviations for any data type or mapping description in this configuration file. Here are some examples to demonstrate the macro feature:

```
[Macros]
NR    = Number
NU    = NumWithUnit
PH    = Phrase
S     = String
TD    = TypeDef
V     = Void
Vis   = V.BookIs<>
I     = S;V;V
NUM   = IDTYPE="NUM"
CAS   = IDCAT="CAS"
CASNUM = IDENT,NUM,CAS
```

Figure 3: Macro example

The first six examples define the aliases for data types. The last three allow the abbreviation I:CASNUM for the CAS number mapping in the example above.

Macros can be defined recursively. In the example above, "CASNUM" is first resolved to "IDENT,NUM,CAS". Then "NUM" and "CAS" are resolved to "IDTYPE="NUM"" and "IDCAT="CAS"".

Even functions and type lists can be hidden behind macros as shown with "I"="S,V,V"="STRING,VOID,VOID" as matching type definition for "CASNUM".

2.2.4.4 Functions

The following functions are provided:

Function	Parameter #	Parameter Meaning	Description
@	1	Table column	Alternative reference to table column. Used only in TypeDef.
CMP	1	Table column	Compare value of referenced column with selection sets and put data field value into matching alternative.

Function	Parameter #	Parameter Meaning	Description
			The table column can be referenced by either the column name from the first row in quotes, or by a column number (beginning with 0). Comparison is case-sensitive.
CMPNC	1	Table column	Same as CMP, but comparison is not case-sensitive.
EXTERM	1	List of separator characters (such as ".,-")	The type parses its value up to the next character found in the list of separator characters. The separator character is excluded from the value.
INTERM	1	List of separator characters	Same as EXTERM, but the separator character is included in the value.
TOTERM	1	List of separator characters	Same as INTERM, but the separator character is left to the following value.
LIST	1	List of separator characters	Reads a list of values into a property with multiple values (such as R phrases). Values are split according to the separator characters.
REGEXP	3	Search expression Opt. expression to replace Opt. Expression to pass to following type	Regular expression (see below)
ADD	1	Table column	The content of the referenced column is appended to the current data value. This may affect subsequent types in the current type list if the current type expression does not handle the entire appended value.
ADDSTR	2	Text opt. Table column	Appends the provided text to the current value. If the second parameter specifies a table column, the text is appended to the value of the table column instead.
ADDTO	1	Table column	Data not being processed by the current type expression is appended to a different column.
INSTIS	2	opt. Table column opt. True-expression	Forces a new characteristic instance depending on a field value.
BOOKIS	2	opt. Table column opt. True-expression	Writes data depending on field value.
Left	2	Length x | "s" opt. Inclusive-Flag	Extracts the first x characters of the current value. Or, extracts everything up to and including (default), or excluding the first occurrence of the comparison string "s"
Mid	4	Position x | "s1" opt. Length y | "s2"	Extracts y characters starting from position x (starting from 0) in the current value.

Function	Parameter #	Meaning	Description
		opt. Inclusive-Flag opt. Inclusive-Flag	Or, extracts everything up to and including (default), or excluding the first occurrence of the comparison string “s2”, starting from and including, or excluding (default) the first occurrence of “s1”.
Right	2	Length x | “s” opt. Inclusive-Flag	Extracts the last x characters of the current value. Or, extracts everything up to and including (default), or excluding the last occurrence of the comparison string “s2”, starting from the end.
Insert	3	Table column Position x | “s” opt. After-Flag	Inserts the content of the referenced table column into the current value at position x. Or, inserts the column content before (default), or after the first occurrence of the comparison string “s” in the current value.
InsStr	3	Text Position x opt. After-Flag	Inserts the given text into the current value on position x. Or, inserts the text before (default), or after the first occurrence of the comparison string “s” in the current value.
InsTo	3	Table column Position x Opt. After-Flag	Inserts the current value into the value of the referenced table column at position x. Or, inserts the current value before (default), or after the first occurrence of the comparison string “s” in the value of the referenced table column.
Upper	4	opt. Pos. x | “s1” opt. Length y | “s2” opt. Inclusive-Flag opt. Inclusive-Flag	Converts all characters in the given range to upper case. The range is defined as described with the right function.
Lower	4	opt. Pos. x | “s1” opt. Length y | “s2” opt. Inclusive-Flag opt. Inclusive-Flag	Converts all characters in the given range to lower case. The range is defined as described with the right function.

Table 41: List of functions

2.2.4.5 Type Details

2.2.4.5.1 Number / NumWithUnit – Numerical Values and Units

A number is separated by any whitespace character (blank, tabulator) from the subsequent value part and can contain the following characters:

Characters (Range)	Description
0 to 9	Digits
. ,	Decimal point. Both period and comma are allowed.
- +	Sign
E	Exponent
<identifier>	Any characters for NUMWITHUNIT

Table 42: Possible values for type number

The syntax of a number is as follows:

Number ::= [<sign>]<integer>[{. | ,}<integer>][E[<sign>]<integer>].

Dot and comma are equally recognized as decimal separator and can be used mixed within an import table. Not allowed are separators for thousands (1,000.00)

The units of the NumWithUnit type offer limited string functionality. All characters can be used but units are always separated from the number and from the subsequent value by whitespace characters.

Numerical values can also be written as ranges following the same rules as in *EHS Management*. For example, two values can be combined with a dash (-) and lower and upper limit can be preceded by operators (>, >=, <, <=). Like > 12.5 - <= 14.6 g/cm3.

2.2.4.5.2 Phrase – Phrase handling

The type phrase offers the same functionality as string.

2.2.4.5.3 String – Texts

Strings are used for any texts. Without any function or regular expression, they read a complete data field and do not leave any value for a potential subsequent type. Therefore, the string type without any additions makes sense only at the end of a type list or as a single element of a type definition.

The functions InTerm, ExTerm and ToTerm allow values of data fields to be split into separate strings. InTerm includes the separator character in the string, ExTerm discards it and ToTerm passes it to the subsequent type.

String is the default type if no type definition at all is provided, or, if a remainder of data elements or value parts is left over from previous types.

String as standard type can even be discarded in the middle of a type list:

“NUMBER;;VOID“ is equal to “NUMBER;STRING;VOID“.

However, functions, regular expressions and selection lists are not possible in this syntax.

2.2.4.5.4 TypeDef – Row-Dependent Targets

With the TypeDef type, targets of data fields can be specified not only by column, but also by row. An auxiliary column defines a target to which a TypeDef column refers.

The data field in the auxiliary column must contain the target description, followed by the type definition, separated by an exclamation mark (!).

Properties must always be followed by the corresponding characteristic category in this case.

Target and type definition refer only to data in the same table row.

The target (second row) of a column referring to such an auxiliary column must be defined as “$T”. The type definition (third row) may contain only the TypeDef type, no further combinations with String or Number, and so on.

TypeDef without function and parameter refers to the preceding column.

With the enhancement TYPEDEF.@<Column>, any other column can be referenced by either the column number (starting with 0), or the column name (first row) in quotes. Column names should be unique for this purpose. If they are not, the left-most matching column is taken.

2.2.4.5.5 Void – The Null Type

This works as a placeholder for target data elements which are only to be initialized. This type does not look for any match in the data field.

The Void type is mainly used with identifiers. Targets like “I:IDTYPE=“NUM“,IDCAT=“CAS“,IDENT“ need a type definition “VOID;VOID;STRING“. To avoid this, the order of target data elements could be reversed: “I:IDENT, IDTYPE=“NUM“,IDCAT=“CAS““.

2.2.4.5.6 Date - Calendar Date

Some characteristics in *SAP EHS Management* require a calendar date in a specific, user-defined format. The *EH&S OCC* can automatically convert dates to the needed format. To use this functionality, the data type DATE is required for such a table column. The date then needs to be provided in the format <YYYYMMDD>. For example, date 20141231 is the 31st of December 2014.

If connected to an SAP system, the *EH&S OCC* will convert the date to the format specified in the user settings of the system. Without connection, the format specified in the offline import settings of the *EH&S OCC* is used.

2.2.4.5.7 Variables

Variables can occur as parameters of functions (see next chapter). They can have any value which the specification import function can process. These are:

Data Type	Description
Void	Empty. Initial type.
Integer	Numbers from –2147483648 to +2147483647.
Double	Floating point with range +/- 1.7E +/- 308 (15 digits).
String	Character string of unlimited size.
Column	Reference to a column within the table (Integer or String).

Table 43: Variables and their types

A variables' type can differ from one function call to the next. It is set whenever the value of the variable is set by a function call (ie. sscanf<>). Uninitialized variables (Voids) are not allowed with Call-By-Value-Calls (ie. sprintf<>). Call-By-Reference-Calls (ie. sscanf<>) on the other hand set their variables as required by the function. Non-critical errors can occur if variables in Call-By-Value-calls have no valid type. In this case, a remark is written to the log file.

2.2.4.6 Function Details

2.2.4.6.1 @ - Reference Columns

See description of TypeDef above.

2.2.4.6.2 CMP / CMPNC – Helper Columns

This function compares data field values with values in helper columns in the same row. It is useful in combination with selection lists.

2.2.4.6.3 ExTerm / InTerm / ToTerm – Scanner for Terminating Symbols

See also the description of regular expressions. A string can be passed as parameter. If, in the data field the character is found, the part of the string terminates for the type with this character.

There are three different scanners available for terminating symbols.

ExTerm excludes the terminating character, while InTerm adds the character to the read part of the text. ToTerm passes the terminating character to the following type.

Without parameters, such as STRING.EXTERM, a selection of standard terminating characters is used. These are the so-called Whitespace-characters.

Whitespace	Description
Control characters	Carriage Return and Linefeed
Spaces	Spaces and Tabulator

Table 44: Whitespace characters

2.2.4.6.4 List - Multiple Value Assignment

The list-function specifies lists of values. As a parameter, the delimiter is expected. In *EHS Management*, the list is a multiple-value assignment.

2.2.4.6.5 RegExp – Regular Expressions

Regular expressions can either be appended in braces behind the type or as parameters of the RegExp function. The syntax is according to the POSIX standard.

Expression		Syntax	Description
Any Character		.	Any character.
Zero or More		*	0 or multiple occurrences of the previous pattern.
One or More		+	1 or multiple occurrences of the previous pattern.
Set of Characters		[]	One of the characters between the brackets A range of characters can be specified by a dash -: like [0-9], [A-Z] or [r-y], etc.
Grouping		()	Groups a sub-expression.
Beginning of Row		^	Matches the beginning of a row.
End of Row		$	Matches the end of a row.
Or		|	Matches either the pattern left or to the right of the Or operator. Is usually used in groups: {red|white}wine
Tagged Expression		{}	Marks a text pattern to be referenced in a replace expression by \N.
Escape		\	Escapes the following character. This way the control characters themselves can be used in expressions.
	Tab Character	\t	Tabulator, Unicode U+0009.
	Carriage Return	\r	Carriage return, Unicode U+000D.
	Line Feed	\n	Line feed, Unicode U+000A.
	Backslash	\\	Backslash.
Nth Tagged Text		\N	In a replace expression, this references the Nth tagged expression of a search pattern. N represents a number between 1 and 9. 0 references to the entire search pattern.

Table 45: Standard patterns in regular expressions

Expression	Syntax	Description
Character Not in Set	[^]	Matches any character not contained in the set.
Prevent Match	~X	Prevents a match of X.
Repeat N Times	^N	Matches exactly N occurrences of the preceding pattern: like “[0-9]^4“ matches a number with 4 digits.
Maximal Zero or More	@	Matches 0 or more occurrences of the preceding pattern taking the maximum number of characters possible.
Maximum of One or More	#	Matches 1 or more occurrences of the preceding pattern taking the maximum number of characters possible.
Alphanumeric Character	:a	Equal to ([a-zA-Z0-9]).
Alphabetic Character	:c	Equal to ([a-zA-Z]).
Decimal Digit	:d	Equal to ([0-9]).
Hexadecimal Digit	:h	Equal to ([0-9a-fA-F]+).
Identifier	:I	Equal to ([a-zA-Z-$][a-zA-Z0-9_$]*-).
Relational Number	:n	Equal to (([0-9]+.[0-9]*)|([0-9]*.[0-9]+)|([0-9]+)).
Quoted String	:q	Equal to (("[~"]*")|('[~']*')).
Alphabetic String	:w	Equal to ([a-zA-Z]+).
Decimal Integer	:z	Equal to ([0-9]+).
Unicode Character	\x#### or \u####	Matches the Unicode character with the four-digit hexadecimal code ####

Table 46: Enhanced patterns in regular expressions

The function RegExp accepts up to 3 parameters:

The 1st parameter is the string that contains the regular expression. This expression is used for every row in the corresponding column.

The 2nd parameter contains the string that specifies what is to be written in each data field. This string can contain references to (partial) expressions from the regular expression.

The 3rd parameter contains a string corresponding to the 2nd parameter. This expression is passed on to the type expression. This makes it possible to combine regular expressions with other types of expressions

References in the 2nd or 3rd parameter can also be grouped with other references. This means that the existence or non-existence of such a reference can influence the processing of a reference-group.

A regular expression for the Text "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ" like...

REGEXP<"\(0.+F\)\(.+Z\)\(.*\)", "Hex digits = \1, G-Z = \2, Rest = \3">

writes "Hex digits = 0123456789ABCDEF, G-Z = GHIJKLMNOPQRSTUVWXYZ, Rest =", while...

REGEXP<"\(0.+F\)\(.+Z\)\(.*\)", "(Hex digits = \1, G-Z = \2, Rest = \3)">

...writes nothing because of the brackets that combine the references to an AND expression and the 3rd reference returns no content. The additional text, such as „Hex digits...“ is included in this selection. Square brackets instead of round brackets would make it an OR- expression.

2.2.4.6.6 Add – Concatenation

With the add-function you can append the content of an as parameter referenced column to the data that is currently being processed. This can also influence the processing of the following types, as the text is appended there if it was not fully processed.

2.2.4.6.7 AddStr – Data Concatenation

AddStr appends a parameter-string to the data that is currently being processed. If as 2nd parameter, a field is referenced, the string will be appended to that field instead.

2.2.4.6.8 AddTo – Passing Values Across Columns

Sometimes, the entire value of a data field is not covered by the type list. Usually, this results in a warning in the log file. By using the AddTo function, these remainders can be passed to another column without causing a warning.

If further type expressions follow, the AddTo function can be used. It passes the remainder for the current type expression as, for example, defined by a separator character in one of the ExTerm, InTerm or ToTerm functions to the other column and leaves the text behind the separator character for the next type expression.

2.2.4.6.9 InstIs – Force Creation of Instances

Characteristic instances are created only for A, M, or N targets. With the function InstIs, the creation of an instance can be forced with other targets, too. This is often required for compositions, which usually do not carry any properties (M targets), but only components mapped to target type L. The same may apply for user-defined texts.

The InstIs function can reference another column. In this case a new instance is only created if the referenced column is not empty or does match the comparison value specified as optional second parameter.

2.2.4.6.10 BookIs – Prevent Writing Data

Similar to the InstIs function, this function also references a helper column to check whether a data value is to be written or not. If the referenced column is empty or does not match the comparison value, the data currently in process is not written to the target, but passed to the following type expression or left as remainder with a warning in the log file.

2.2.4.6.11 Left, Mid, Right – Position-Dependent Data Extraction

With type function Left<x,i>, the first x characters of the currently processed text can be extracted, with Mid<x1,x2,i1,i2> x2 characters from x1 on. Function Right<x,i> extracts the last x characters.The parameters i, i1 and i2 are flags, that each relate to the last characters of the text. They can have the following values:

FALSE Number = 0
Text = empty = ""
...means that the given positions and lengths describe the text before the characters that are to be extracted.

TRUE Number <>0
Text = not empty = not ""
...means that the given positions and lengths include the characters that are to be extracted.

The so-called inclusive-flags are by default FALSE for positions and TRUE for lengths.

A position can be a number or a text (in quotation marks). If a text is used, the currently processed text will be searched for the term and its start (or end in case of function Right<>) taken as position. If the term cannot be found, the position is –1 which is the end of the text (or the start in case of function Right<>).

2.2.4.6.12 Insert, InsStr, InsTo – Inserting Data and Constants

The function Insert<c,p,a> adds the content of field c at position p to the currently processed text. Flag a controls if the content is added behind (a=1) or before (without 3rd parameter) the position.

Instead of a field content, the function InsStr<t,p,a> inserts a constant text t (within quotation marks). InsTo<c,p,a> inserts the content of the current field at position p of field c. The 3rd parameter a is optional and FALSE by default, which means that the text will be inserted before position p.

Otherwise, the meaning of the position and flag parameters is the same as with the functions Left<>, Mid<> or Right<>.

2.2.4.6.13 Upper, Lower – Changing the Capitalization of Texts

Upper<> changes all lower case letters to upper case letters. Lower<> does the opposite.

Both functions can be given the same ranges as with Mid<> to limit the conversion to a certain part of the text.

2.2.4.6.14 CharToHex, HexToChar – Converting Characters

CharToHex<> converts a text to hex, based on the ASCII-values of the respective characters.

Ie: The text „Hello“ would be converted to „48656c6c6f“. HexToChar<> does the opposite.

Both functions can be given the same ranges as with Mid<> to limit the conversion to a certain part of the text.

2.2.4.6.15 Trim, TrimLeft, TrimRight – Removing Whitespaces

TrimLeft<> removes leading whitespaces, TrimRight<> trailing whitespaces. Trim<> does both.

2.2.4.6.16 SscanF – Reading Characters in a Certain Format

With this function, which is known from ANSI C/C++, it is possible to read in certain parts of the text that is currently in process. Those parts can be saved to variables that can in turn be used later with function sprintf<> to write them back into a text. The parameter list starts with a compulsory parameter that describes the format of the text. At least one variable follows as parameter. The values of the variables are set if a place-holder was defined in the format-text from the 1st parameter and the currently processed text has a match for that place-holder. The syntax of the format-text is:

```
Format-text          ::= <Format-term><Format-text>.
Format-term          ::= <Character string> | <Place-holder>.
Character string     ::= Alphanumeric String.
Place-holder         ::= %[*][<Width>]<Type>.
Width  ::= <Number>.
Type   ::= Character out of {cdiouxeEfgGs}.
Number ::= Integer from N+ \0.
Legend:
  Left ::= Right              Key on the left side consists of expressions on the right side.
  X | A   Either X or A.
  { Expression }              Scope of OR-expression.
  [Expression]                Expression is optional.
  .         End of expression. May be followed by a comment.
  < Expression >              Expression is a key which is resolved further below.
  Text    Self-explaining Text.
   Characters in bold and non-italics are  control characters.
```

Figure 4: Syntax of the format-text for sscanf<>

A format-text can either be a character string or a format-term. With a character string, you can synchronize an input text with the format-arguments. In this way, the function sscanf<> knows where to look for the values for the respective variables in the text.

The format-argument is represented by a placeholder. It always starts with a %-character. A %% is a single %-character for the text and not a place-holder.

To limit the range of the text it is possible to enter an optional width. This value reflects the number of characters available for the placeholder.

An asterisk (*) right behind the % deactivates this argument.

One of the following types must always be added to the %-character:

Format type	Description
c	Represents a character. If a width is given, this can also be a character string.
d	Represents integers with base 10.
i	Represents integers with base 8 (octal), 10 (decimal) and 16 (hex).
o	Represents integers with base 8 (octal).
u	Represents natural decimal numbers from N+
x	Represents integers with base 16 (hex).
e, E, f, g, G	Represent real numbers.
s	Represents a string (text). A given width specifies how many characters should be assigned. A whitespace character always terminates the string.

Table 47: Format types for sscanf<>, printf<> and sprintf<>

Example

sscanf < “%d.%d.%d“ , Day , Month , Year >

Value from table field as input for sscanf = 14.8.2002

The variables Day, Month and Year are set to...

Day = 14

Month = 8

Year = 2002

If the date has a different structure (such as, 14/08/2002), sscanf<> would abort, because the text pattern does not match the format-text (slashes / instead of periods). In this case the variables would keep their initial values.

2.2.4.6.17 Sprintf, PrintF – Generating Formatted Text

Both functions format the values of their given variables according to what is specified in the first parameter which contains the format-text. The usage is similar to function scanf<> as sprintf<> and printf<> are its counterpart.

Sprintf<> generates a new text in the currently processed buffer, whereas printf<> writes the value to the log file as information.

The syntax of the format-text is the following:

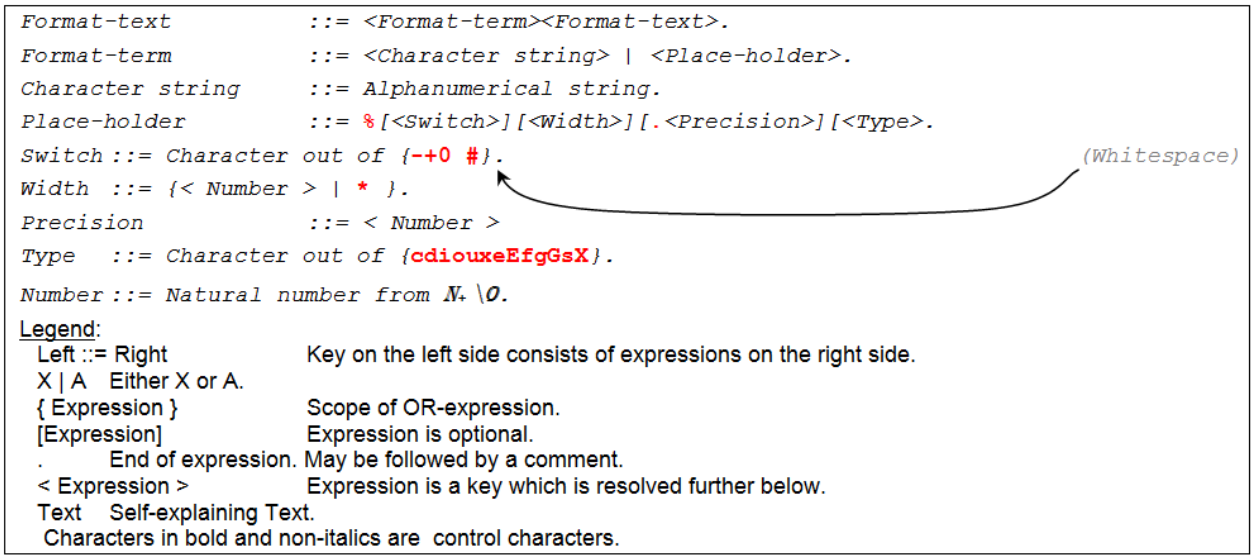


Figure 5: Syntax of a format-text for sprintf<> and printf<>

As with sscanf<>, a format-text can either be a character string or a format-term. With a character string, you can synchronize an input text with the format-arguments. In this way, the functions sprintf<> and printf<> know where to look for the values of the respective variables in the text.

The format-argument is represented by a placeholder. It always starts with a %-character. A %% is a single %-character for the text and not a placeholder.

To limit the range of the text, it is possible to enter an optional width. This value reflects the number of characters available for the place-holder.

The switch configures how leading characters are handled if the width is bigger than required for the value of the respective variable. If it is set to „-“, the value will be left-aligned and filled up with trailing spaces. By default, all values are printed right-aligned. If set to „+“, all numerical values (%d, %i, %u, %e, $E, %f, %g and %G) will be printed with algebraic signs, even if the value is positive. By default the sign is only printed if the number is negative.

If set to „0“ or a whitespace, the value will be filled up with leading zeroes or spaces respectively.

With character „#“, a "0" in case of octal and an „x“ or „X“ in case of a hex formats (%o, %x and %X), will be inserted before the value. With floats (%e, %E and %f), a dot will always be written as decimal delimiter even if it is not necessary. With formats %g and %G even trailing zeroes are kept. For any other type this switch has no meaning.

The precision of a float can be configured with the equally-named field. This value is ignored for character outputs (%c). With ordinal numbers (%d, %i, %u, %o, %x and %X) this is the width including leading zeroes. With floats %e and %E this value specifies the number of digits behind the decimal delimiter including trailing zeroes. With float (%f) the value will be rounded to that digit. With character strings (%s) this value is the maximum number of characters that will be printed.

Example

sprintf < “Date: %02d.%02d.%04d“ , 14 , 8 , 2002 >

The text would then contain...

„Date: 14.08.2002“.

The function sprintf<> prints all standard characters (such as „Date: “ and the dots between the placeholders) into the buffer and fills the placeholders with the formatted values of the given parameters (or constants).

2.2.4.6.18 AddIf – Conditional Appending of Characters

Variables can be used for more than just the buffer. They can also serve as a switch. The function AddIf<> appends based on the first parameter either the second or the third parameter to the currently processed text.

If the first parameter is a "1" or filled with a text, either the value of the second parameter is taken or the third. If, however, there are more than 3 parameters, the third parameter again serves as a flag to switch between the next parameters and so on.

Example

AddIf < 1, „True“ >

...appends the value „True“.

AddIf < 0, „True“ >

...appends nothing.

AddIf < 0, „True“, „False“ >

...appends the value „False“.

AddIf < 0, „True“, 1, „False-True“ >

...appends the value „False-True“.

AddIf < 0, „True“, 0, „False-True“ >

...appends nothing.

AddIf < 0, „True“, 0, „False-True“ , „False-False“ >

...appends the value „False-False“.

...etc...

All parameter can be variables.

Example

AddIf < Condition, „True“ > Condition = „This is some text“

...appends the value „True“ because the text in variable < Condition > is not empty.

2.2.4.6.19 Control Characters

The following control characters have a specific meaning in the first three rows and a TypeDef auxiliary column of an import table:

Character	Description
“	Quotes include strings
,	Comma separates elements of lists and data elements in targets No special function within quoted strings.
.	Period separates the function name from the type. No special function within quoted strings.
;	Semicolon separates characteristics and types. No special function within quoted strings.
[ ]	Square brackets enfold selection sets. No special function within quoted strings.
( )	Round brackets enfold optional types. No special function within quoted strings.
{ }	Curly brackets enfold regular expressions. No special function within quoted strings.
< >	Greater than-/smaller than-characters enfold the list of parameters of a function call. No special function within quoted strings.
=	Initializing in the target field. No special function within quoted strings.
:	Colon separates the characteristics and the property header. No special function within quoted strings.
$	A dollar sign specifies controls in the target fields. No special function within quoted strings.
Tab	Tab stops are prohibited, because of tables in text format.
\	Backslash serves as escape-character for strings.

Table 48: Control characters in the first three header rows

There is no limitation on the usage of the characters A..Z, a..z, umlauts, digits 0..9, underscore _, minus/dash -, plus +, at @, exclamation mark !, slash / etc. Quotation marks and backslashes must be escaped with \ ie. \“ or \\. The field identifier does not need to be put between quotation marks.

2.3 Additional Information Required for Importing Specifications

Importing data into *Specification Management* requires some information that is not definable in the specification import file. This information can be stored in a separate configuration file (.ini).

	`[General]`
Character set	`SC = ISO-Latin 1`
ID	`ID = EHS`
Version	`V = 2.2B`
Date	`D = 19990311`
Date of validity	`VD = 19990311`
Source language	`SL = E`
Date format	`DF = DD.MM.YYYY`
Decimal point or comma	`DN = .`
	`[Custom]`
	`ID = 0000010200`
	`NAME = My Company`
	`CITY = Amsterdam`
	`CNTY = NL`
	`[PhraseCat]`
Phrase catalog	`ID = SIGMA`
Catalog version	`V = 1.0`
Catalog date	`D = 19980819`
	`...`

Figure 6: Configuration file

This header information is appended to all specifications of the import file.

Alternatively, this header information can be entered in the *EH&S Open Content Connector (EH&S OCC)* under *Data Import -> Settings*. However, macros can be defined only in a configuration file.

3 Preparing Data for Phrase Import

To load phrases into *Phrase Management*, you must provide data in a table-based format, for example, a relational database, Microsoft Excel file, or a text file.

In most cases, the data is exported from some external software system or database into a table-oriented file. This file can have to following format:

CATPIN	PHRGRP	PHRID	...
CAT01	GRP01	ID01	...
		ID02	...
	GRP02	ID01	...
...	...	...	...

Table 49: Structure of phrase import file

The data has to be provided in separate columns with a specific sequence. A header row has to be added to map the data to the fields in *Phrase Management*. The syntax of that row is described in the following chapter.

After mapping the data to fields in the specification database, also referred to as data elements with correct syntax and semantic logic, the phrases can be imported.

> **Note**
>
> The *EH&S OCC* needs the data in a Unicode text file format to import the phrases. Therefore, you need to convert all other file formats to Unicode format with the table columns separated by tabulators.
> After the conversion, you must change the file extension to PIT before starting the phrase import function to generate a data transfer file for import.

3.1 Structure of the Phrase Import File

To describe the data format for the phrase import, you have to add a header rows to your data. Each field of a header row relates to the data in the same data column. This row describes the content of the column, as well as whether it is an interface between logical column descriptions and the phrase structures in *Phrase Management*.

The phrase data that you wish to import starts in the second row.

The first column has a key meaning, and is used to separate phrase libraries, even if the phrase import does not support more than one phrase library in one transfer file. It is mandatory for the phrase import algorithms.

In most cases, the first column is filled with only one field in the second row, which is the first data field. It holds the CATPIN that specifies the library identifier in *Phrase Management*. All fields beneath this field can be left blank or repeated with the same value to specify in the phrase import that the subsequent rows belong to the same library.

The first row contains the captions for the columns. These captions are used to link to the macros in the configuration file (such as “header.ini”). The following macros are predefined:

Macro|Field Length		Description | Corresponding Tables in the Phrase Database	
CATPIN	5	Identifier of phrase library	TCG61
CATDIST	40	Distributor of phrase library	
CATDATE	8	Creation date of phrase library (Format = JJJJMMTT)	
CATVERS	10	Version number of phrase library	
NUMRNGE	10	Name of number range object for identification of phrase library.	
CATREM	60	Remark on phrase library	
CATNAM	40	Description of phrase library	TCG62
CATLANGU	1	Language key of library description	
PHRGRP	10	Identifier of phrase group	TCG63
PHRGREM	60	Remark on phrase group	
PHRGNAM	40	Description of phrase group	TCG64
PHRGNL	1	Language key of phrase group description	
PHRID (KEY)	15	Phrase key	ESTPH
SRLANGU (SRL)	1	Key of original language	
REM	60	Remark on phrase header	
PHRCODE (CODE)	40	Phrase code	ESTPP
LANGU (LC)	1	Key of phrase language	
PHRTEXT (PHRASE)	132	Phrase text	
PHRGRAPH (GRAPHIC)	30	File name of phrase graphic	
PHRREM	60	Remark on phrase item	
PHROFLG	1	Transfer original text	ESTPO
OPHRID	15	Key of original phrase	
OCATPIN	5	Identifier of original phrase library	
JPHRID	15	Phrase key for phrase join	ESTPJ
PHRSEL	10	Phrase selection (name of phrase set)	
ORD	4	Order number or phrase assignment	

Table 50: Predefined macros

To define new macros, it is necessary to edit the configuration file. One section (the one in brackets [ ]) in this file represents a field subscription and the name of a macro respectively. It is possible to define three values in this section. The fields “DSTRUCT” and “TSTRUCT” describe the method and target of reading and booking of the data in the sheet to the phrase library structures.

For more information, see chapter 6.1

The phrase (library) database in *Phrase Management* consists of several tables. In eight of these tables, the phrase import function can import data.
The following figure shows the connections between these tables, the join of the dedicated macros and the tables in the phrase database:

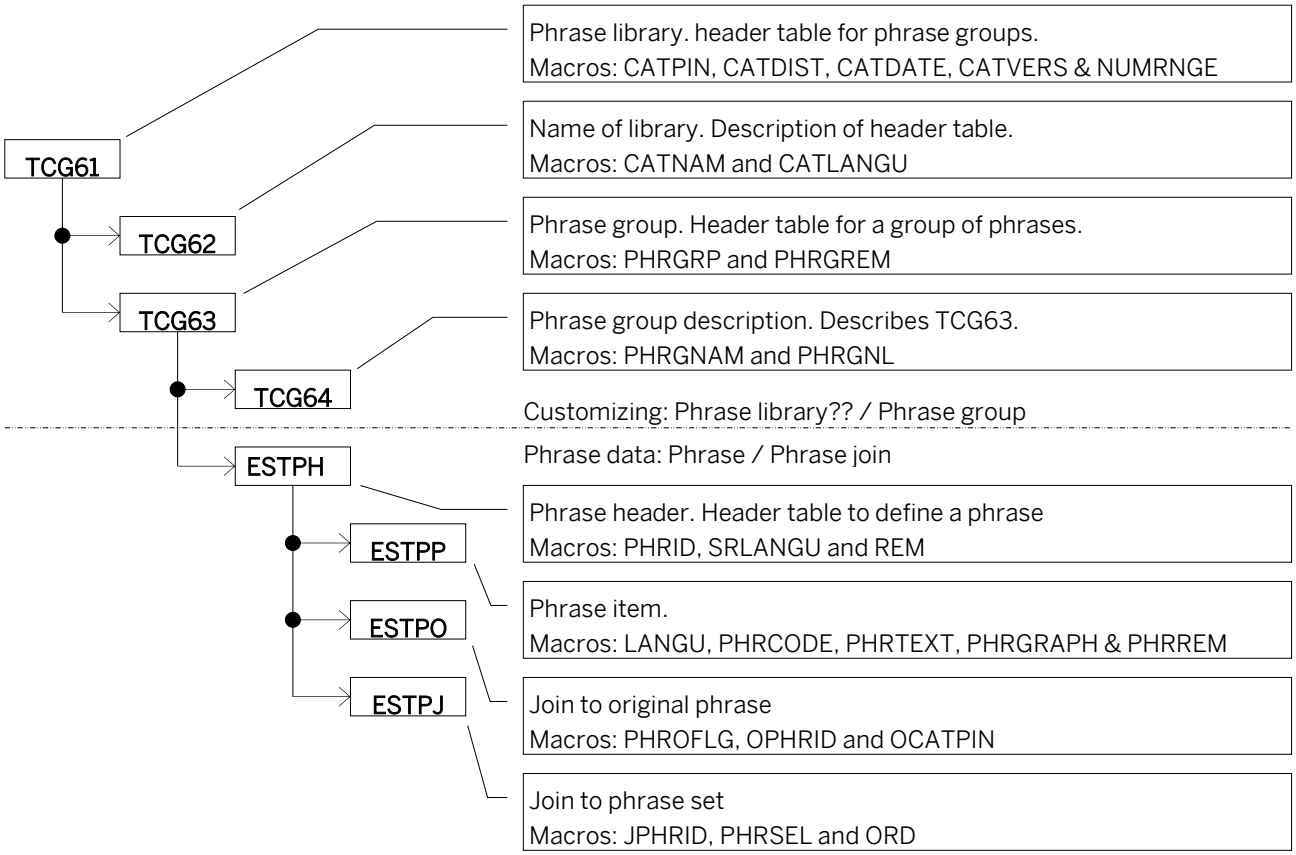


Figure 7: Import tables of the EHS Management phrase database

CATPIN	CATDIST	CATDATE	CATVERS	PHRGRP	KEY	SRL	LC	CODE	PHRASE	GRAPHIC
CAT01	Phrase distributor 1	20010725	1.0	GRP01	PHR01	E	E	C1	English Text 1	...
							D	C1	German Text 1	
					PHR02	E	E	C2	English Text 2	
							D	C2	German Text 2	
					PHR03	E	E	C3	English Text 3	
				GRP02	PHR02	E	E	C2	English Text 2	
							F	C2	French Text 2	
					PHR04	E	E	C4	English Text 4	
				GRP03	PHR05	E	E	C5	English Text 5	
					PHR06	E	E	C6	English Text 6	
				GRP04	PHR01	D	D	C1	German Text 1	
					PHR02	D	D	C2	German Text 2	
					PHR03	D	D	C3	German Text 3	
				GRP05	PHR01	D	D	C1	German Text 1	
					PHR02	D	D	C2	German Text 2	
							R	C2	Russian Text 2	
				GRP06	PHR03	D	D	C3	German Text 3	

Table: Example with one phrase library

The first column contains the key of the phrase library. Usually, this is the name of the phrase library (CATPIN). The hierarchic order is important to bind all data to its superior object. The second, third, and fourth columns specify the distributor of the phrase library, its publication date, and its version. The fifth column groups the data by phrase group and the sixth and seventh column contains the phrase key and its source language.

The rest of the columns contain the language of the specific phrase, its code, the text, and an optional path to a symbol that have to exist in the phrase database.

3.2 Additional Information Required for Importing Phrases

Importing data into *Phrase Management* requires some information which is not definable in the import file. This information can be stored in a separate configuration file (.ini).

	[General]
Character set	SC = ISO-Latin 1
ID	ID = EHS
Version	V = 2.7B
Date	D = 19990311
Date of validity	VD = 19990311
	[Macros]
	NR = Number
	NU = NumWithUnit
	PH = Phrase
	S = String
	TD = TypeDef
	V = Void
	[CATPIN]
	NSTRUCT = Catalog.ID
	DSTRUCT = 1:CATPIN
	TSTRUCT = string
	[CATDIST]
	NSTRUCT = Catalog.Distributor
	DSTRUCT = 1:CATDIST
	TSTRUCT = string

Figure 8: Configuration file

This header information is appended to all phrases of the import file.
Alternatively, this header information can be entered in the *EH&S Open Content Connector (EH&S OCC)* under *Data Import -> Settings*. However, macros can be defined only in a configuration file.

4 Data Import

4.1 Specification Import

There are two different ways to use the specificationiImport function in *EH&S Open Content Connector*:

- Generate a data transfer file:
  Without connection to an SAP system, the specification import function of *EH&S OCC* uses the provided import data to generate a data transfer file. You can use the import specifications transaction (CG33) to import this transfer file into *Specification Management* (EHS-BD-SPE).
  To generate a transfer file, proceed as follows:
  1. If connected, disconnect from the *SAP EHS Management* system.
  2. Open the specification import function under *Data Import -> Specification Import* without reconnecting to the SAP System.
  3. Upload the import file with the specification data you wish to import.
     Note that the file must be in pure ASCII or Unicode text format.
  4. Choose either to create a separate transfer file for each specification of the import file or one single transfer file to contain all specifications.
  5. The import function parses the data in the import file and creates one or multiple specification transfer files.
- Import data directly:
  With connection to an SAP system, the specification import function of *EH&S OCC* can import data directly into the specification database by using the standard SAP BAPI interface of *EHS Management*
  To import data directly, proceed as follows:
  1. Connect to an *SAP EHS Management* system.
  2. Open the specification import function under *Data Import -> Specification Import*.
  3. Upload the import file with specification data you wish to import.
     Note that the file must be in pure ASCII or Unicode text format.
  4. The import function parses the data file and calls the appropriate BAPIs to save the imported data in the SAP system.

The parsing process is visualized with a progress bar that also displays the elapsed time and estimated rest duration. You can stop the importing process after the import of any specification.

A detailed log of the parsing and importing process is written to a log file.

4.2 Phrase Import

There is only one way to use the phrase import function in *EH&S Open Content Connector*: You can generate a transfer file and then use the import phrases transaction (CG31) to import this transfer file into *Phrase Management* (EHS-BD-PHR).

To generate a transfer file, proceed as follows:

1. Open the phrase import function under *Data Import -> Phrase Import.*
2. Upload the import file with phrases you wish to import.
   Note that the file must be in pure ASCII or Unicode text format.
3. The import function parses the data in the import file and creates a phrase transfer file.

A detailed log of the parsing and importing process is written to a log file.

4.3 Import Process

A progress bar visualizes the status of the import process and the current actions.

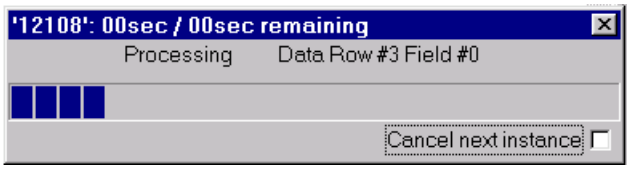


Figure 9: Progress bar

The title bar displays the name of the specification or the phrase library being processed in single quotes. The time elapsed and the estimated remaining time is shown after the colon. The first dialog row contains information about the current action. “Loading” means that a row is currently being loaded. “Processing” means that the row is being parsed. The row and column number is also shown. The blue bar indicates the progress.

The “Cancel next instance” checkbox allows you to cancel the import process once the current specification or phrase library has been processed.

4.4 The Log or Protocol

The specification import function logs all its actions. The log contains general and important information, as well as warnings and errors.

In addition, there is a chronological log regarding the sequence of actions and occurring errors or warnings, together with their exact context (row and column).

The log file with the description SIT.LOG is created in a configurable folder.

A log entry has the following structure:

<day>/<month>/<year> <hour>:<minute>:<second>

SIT_ID(<hex number>)/<type>: <header>

<information>

The SIT_ID is an internal number referencing the internal program code and relevant only for developers. The type of information can be “Info“, “Warning“, “Error“ or “Fatal Error“.

Info contains information about the process, such as a time stamp or information about opening and closing an import table.

Warnings are logged if a required target or type information was not complete and has been automatically filled, or if data could not be mapped to data elements and is left over.

Errors are divided into non-critical errors that do not affect the general import process, and fatal errors that cause the import process to end.

5 Settings

You can configure the specification and phrase import functions under *Data Import -> Settings*.

5.1 General

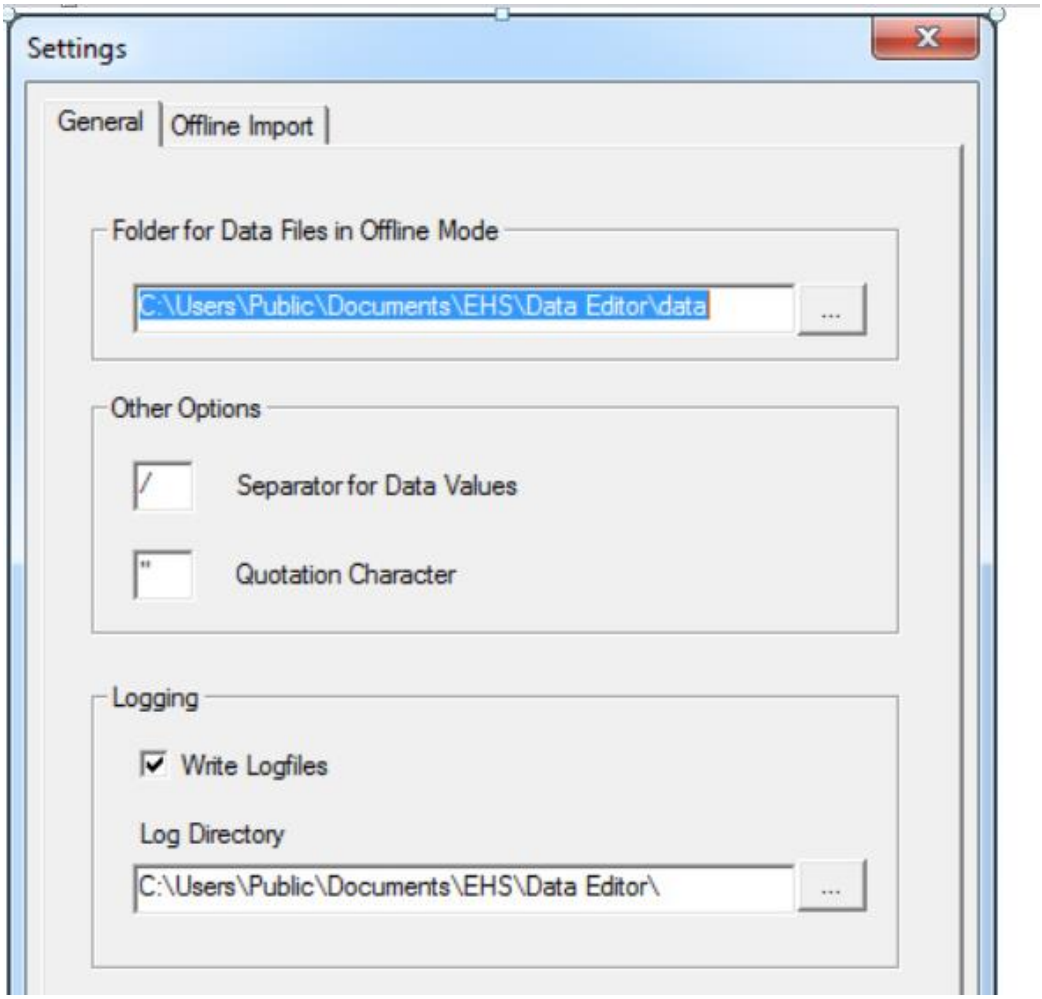


Figure 10: General settings

In section *Location of Specification Data Files in Offline Mode* in the *general* settings dialog, you can configure the path for output files of the offline specification or phrase import.

The *Separator for Data Values* defines a character that is used to separate single data values within one string. This character is defaulted to '/'. The separator should never occur in one of the data values.

With the *Quotation Character,* you can define a character which marks a quoted section within a specification or phrase import file. The text enclosed in this character is loaded into the same field, regardless of its content. This is necessary if you wish to load field values including special characters, such as the tabulator character. The quotation character should never occur in one of the data values. The default value of this field is the double quotation mark (").

In the *Log Directory* section you can configure the path for log files. To switch on the logging, you need to check *Write Logfiles*.

To change the language of the user interface, choose *Options* -> *Preferences* and click on the *Systems* tab. Choose the language file at option *LanguageFile*.

> **Note**
>
> SAP delivers the EH&S Open Content Connector in English and German only. However, you can add your own languages.

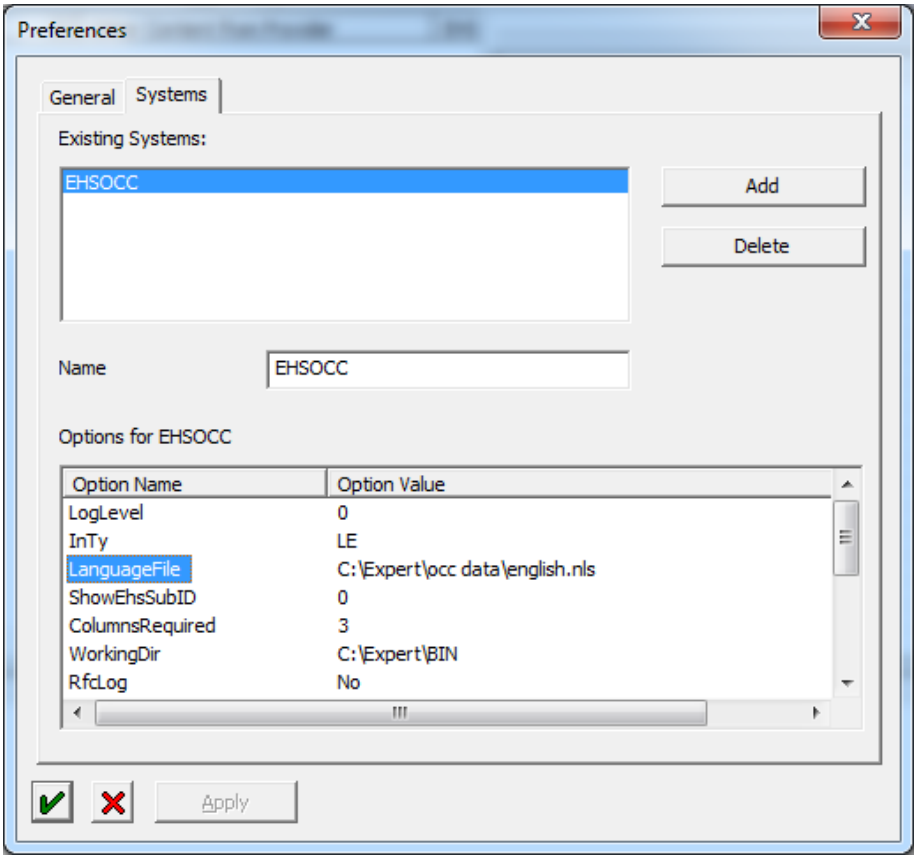


Figure 11: Language file

5.2 Offline Specification Import

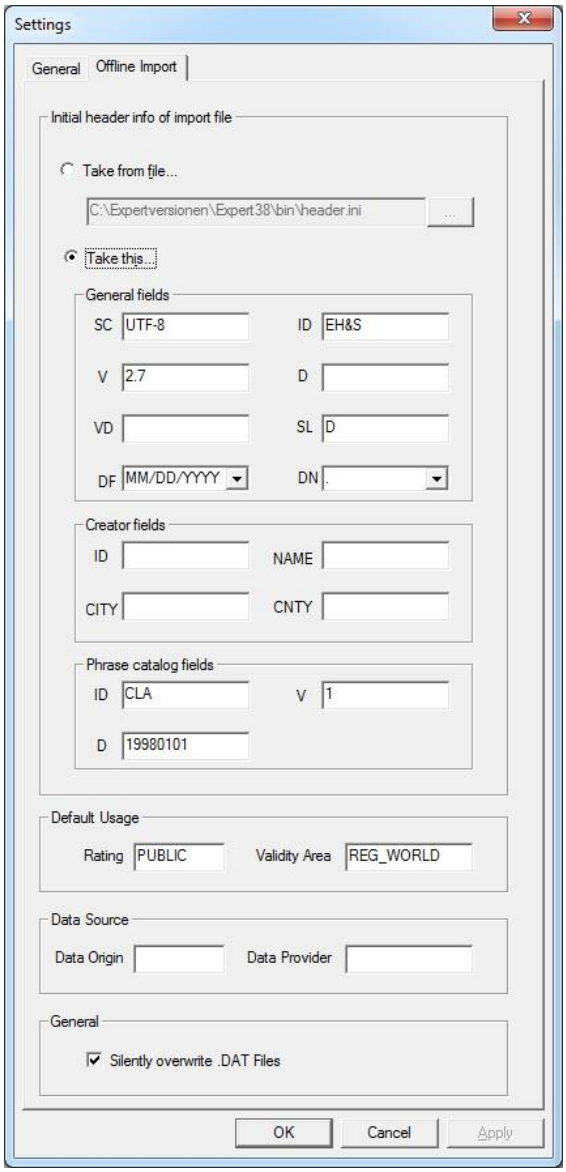


Figure 12: Settings for the specification import

In Initial header info of import file, you define the header information for the transfer files generated by the phrase or specification import function.

The optional and edit field *Take from file* allows you to specify a configuration file for the specification import function. An exemplary .ini file named "header.ini" is included.

> **Note**
>
> Because of further declarations included in this file, it is vital to configure the correct path here, even if this option is not active.

Alternatively the header information can be entered in the subsequent input fields.

For further information, see the documentation in the Help Portal under http://help.sap.com/ehs27. Choose your release, go to section *SAP Library*. Open the application help documentation and go to *Basic Data and Tools (EHS-BD) -> Tools (EHS-BD-TLS) -> Import and Export -> Import: Process*.

In the *Default Usage* section, a usage can be configured that will be taken if no usage is specified in the import table.

In the *Data Source* section, a default *Data Origin* and *Data Provider* can be configured that will be taken if no values are specified in the import table.

If the *Silently overwrite .DAT Files* box is selected, no warning will be issued if a .dat file with the same name already exists.

6 Appendix

6.1 Macros

It is possible to define the entire header information for a column in a macro. For example, column name, target and type definition, usually split into three header rows, can be combined in the very first row.

The syntax for a definition macro in the first header row is as follows:

```
header ::= <section> [<parameter list>].
section              ::= section name from configuration file.
parameter list       ::= <parameter> [<parameter list>].
parameter            ::= /<parameter type>[:<replacement>].
parameter type       ::= NR | DR | TR
replacement          ::= <string>=<string>
string ::= alphanumeric text in quotes.
Legend:
  left ::= right              Key on the left side consists of expressions on the right side.
  X | A   Either X or A.
  { Expression }              Scope of OR expression.
  [Expression ]               Optional expression.
  .       End of expression. May be followed by a comment.
  <expression>                Expression is a key which is resolved further below.
  Text    Self-explaining text.
  Characters in bold and non-italics are control characters.
```

Figure 13: Syntax of definition macros

Section references a section in the *EH&S OCC* configuration file. If the section is not found, there is no macro replacement and target and type definitions are expected in the second and third row. If the section is found, the following entries are looked for:

NSTRUCT Column name, used as reference for like CMP and BookIs functions.

DSTRUCT Target.

TSTRUCT Type definition.

All entries are optional. DSTRUCT does not overwrite an existing target. TSTRUCT does not overwrite an existing type definition. Both entries are ignored then.

In addition all entries can get parameters to make the macros more dynamic:

NR Replacement parameters for the value of the NSTRUCT entry.

DR Replacement parameters for the value of the DSTRUCT entry.

TR Replacement parameters for the value of the TSTRUCT entry.

After the parameter type and colon a search/replace expression must be defined as “search”=”replace". Any occurrence of “search” in the value from the configuration file is replaced by “replace”. There can be multiple parameters of the same type.

6.2 Examples

6.2.1 Optimizing:

This example demonstrates how header information and data can be reduced to a minimum.

It is, for example, not required to repeat the specification ID and the CAS number in every row. The specification import function automatically adds a data row to the previous specification as long as the content of the first column is empty or does not change. The same applies to the CAS number.

Leaving data fields empty actually improves the performance of the specification import function as it avoids needless processing of values.

SubID	CAS-Number	Language	Product name	Density	
H:SubID	I:IDTYPE="NUM", IDCAT="CAS",IDENT	I:LANGU	I:IDENT,IDTYPE="NAM", IDCAT="PROD"	M:SAP_EHS_1013_005_VALUE; SAP_EHS_1013_005	
	VOID;VOID			NUMBER	
12108	50-00-0	DE	Formaldehyd ...%	0,855	Specification 12108
		DK	formaldehyd ...%		
		EN	formaldehyde ...%	1,65	
		ES	formaldehído ...%	0,915	
		FR	formaldéhyde ...%; aldéhyde formique ....%	1,65	
		IT	formaldeide ...%; aldeide formica ...%	1,19	
		NL	formaldehyde ...%	2,69	
		PT	formaldeído ...%	0,995	
2	50-21-5	DE	Milchsäure	2,434	Specification 2
		DK	maelkesyre	1,11	
		EN	lactic acid	3,2	
		ES	ácido lactico	0,86	
		FR	acide lactique	1,805	
		IT	acido lattico	1	
		NL	melkzuur	1	
		PT	ácido lactico	1	
20720	52-51-7	DE	Bronopol	1	Specification 20720
		DK	bronopol	1	
		EN	bronopol	1	
		ES	bronopol	1	
		FR	bronopol	1,2	
		IT	bronopolo	1,2	
		NL	bronopol	0,86	

Table 51: Optimizing example

6.2.2 Identifiers

6.2.2.1 Using of Language-Dependent Identifiers

If you wish to specify a usage for a certain language-dependent identifier, you can either add the language to the mapping or assign the usage to the identifiers dynamically. For this, the usage column must follow the I:IDENT column directly and must not specify a record.

Type	Category	Langu	Ident	Rating	Validity Area
I:IDTYPE	I:IDCAT	I:LANGU	I:IDENT	U:VACLID	U:RVLID
STRING	STRING	STRING	STRING	STRING	STRING
NAM	PROD	DE	Ident1	PUBLIC	DE
NAM	SYN	EN	Ident2	PUBLIC	EN
NAM	ANNEXI	FR	Ident3	PUBLIC	FR

Table 52: Example: Language-dependent identifiers

6.2.2.2 Identifiers with Regulatory Lists

This example shows how a regulatory list can be added to an identifier:

Identifier
I:IDENT,SUBLIST,IDTYPE="NAM",IDCAT="PROD",LANGU="DE"
STRING.exterm<";">; STRING
IdentifierXY;ADNR

Table 53: Example: Identifiers with regulatory lists

6.2.3 Additional Information

6.2.3.1 Usage

Additional information such as usages, sources, user-defined texts, etc. cannot create value assignment instances on their own. They usually relate to an instance created by a characteristic which has to be created before the additional information.

Density	Usage
M:SAP_EHS_1013_005_VALUE;SAP_EHS_1013_005	U:VACLID,RVLID
NUMBER	STRING.EXTERM<";"">
12	PUBLIC;DE
30	PUBLIC;REG_WORLD

Table 54: Example: Usage

6.2.3.2 User-Defined Texts

User-defined texts can be assigned with mapping letter F:

Density	Category	Langu	Text
M:SAP_EHS_1013_005_VALUE;SAP_EHS_1013_005	F:TEXTCAT	F:LANGU	F:TEXT
NUMBER	STRING	STRING	STRING
12	RM	DE	Text1

Table 55: Example: User-defined texts

6.2.3.3 Assessment

Assessments can be assigned by mapping letter R. The only possible target field is the assessment itself.

Density	Assessment
M:SAP_EHS_1013_005_VALUE;SAP_EHS_1013_005	R:RELID
NUMBER	STRING
12	3
30	2

Table 56: Example: Assessment

6.2.3.4 Additional Information Without Instance

Sometimes it is necessary to write additional information that does not relate to a characteristic. In this case a new instance has to be explicitly created which can be done with mapping letter A.

Sort Sequence	Text
A:ORD;SAP_EHS_1011_001	F:TEXT,TEXTCAT="RM"
STRING	STRING
1	TextABC

Table 57: Example: Additional information without instance

6.2.3.5 Restrictions

A specification restriction can be created with a usage that relates to the specification header.

Specification Header	Restriction
H:SUBID,SUBCAT="REAL_SUB",AUTHGRP="ALL"	U:VACLID,RVLID;H:SUBID
STRING	STRING
SpecificationXY	PUBLIC;REG_WORLD

Table 58: Example: Restriction

6.2.4 Compositions

In this example a component is written to the standard composition.

Composition	Sort Sequence	Category	Average
L:SUBID;SAP_EHS_1012_003	L:ORD	L:COMPCAT	L:COMPAVG
STRING	NUMBER	STRING	NUMBER
ComponentXY	1	ADDITIVE	20

Table 59: Example: Composition

6.2.5 DG Data

6.2.5.1 Transport Classification

Use mapping letter f:

Regulatory List	Specification	DPot	No HM Full	No HM Empty	Status
f:LWDG;SAP_EHS_1022_023	f:SUBID	f:DPOT	f:NHM	f:NHME	f:WOS
STRING	STRING	STRING	STRING	STRING	STRING
ADR	UNSUB001	(,)			10
CFR	UNSUB001	(,)			1

Table 60: Example: Transport classification

6.3 Frequently-Asked Questions

6.3.1 File Format

The input format for a specification import file is a text file. Table columns must be separated by tabulators. Tabulators may not be used for any other purpose. Rows are separated by carriage return / line feed. Blanks / spaces are part of the data fields.

Data can contain data from various code pages for like identifiers and user-defined texts. The appropriate language-dependent conversions are done when uploading into *EHS Management*. However, Microsoft Excel does not offer proper export functionality for non-standard code pages. SAP recommends you save the file as "Unicode text (*.txt)", or as "UTF-8".

To use Unicode only languages, you need to use Unicode file format, even if all contained data conforms to Latin 1 code page.

6.3.2 Import Header Information

The necessary header information for generating a valid import file can be stored in the *EH&S OCC* configuration file. See chapter 2.3.

The information can also be specified in the *EH&S OCC* under *Data Import -> Settings dialog*.

6.3.3 Definition- and Data File

The file starts with three rows describing the data format of the file:

Remark / description of the column. Required for cross-referencing between columns.

This is the mapping row specifying the target fields in *EHS Management*.

Data Type row.

6.3.4 Multiple Characteristic Instances

Multiple instances for a characteristic have to be split into multiple table rows in the import file.

6.3.5 Multiple Values for a Property

Multiple values for a property or additional information can be handled by the LIST function. In this case, all values for the property are expected in a single table field, separated by some separator characters. For example, STRING.LIST<","> looks for commas and separates a field content into multiple values.

For additional information such as usage/validity area, it is also possible to keep multiple values in multiple table rows.

6.3.6 Compositions, References, Transport Classification

These elements of the EHS data model require referencing other specifications (as components or data references).

In an online mode, a reference to another specification is done either by specification id or a set of unique identifiers. A component (target type L) can be referenced for example, with the target "L:SUBID". Or alternatively "L:IDENT,IDCAT="CAS",IDTYPE="NUM"" to reference by CAS number.

Unlike the offline mode, it is necessary to provide the specification ID and identifier of a reference because according to the import file syntax (file extension .dat) for each specification at least one identifier is mandatory.

It works similar for reference specifications (target type D) and references to UN list specifications in the transport classification (target type f).

Target types are case-sensitive.

6.3.7 Values

The definition of the value structure is specified win the corresponding data type field.

Texts should be defined as data type String.

Numbers should be defined as data type NUMBER.

Numbers with units should be defined as data type NUMWITHUNIT.

Phrases should be defined as data type String.

References are handled as text / string.

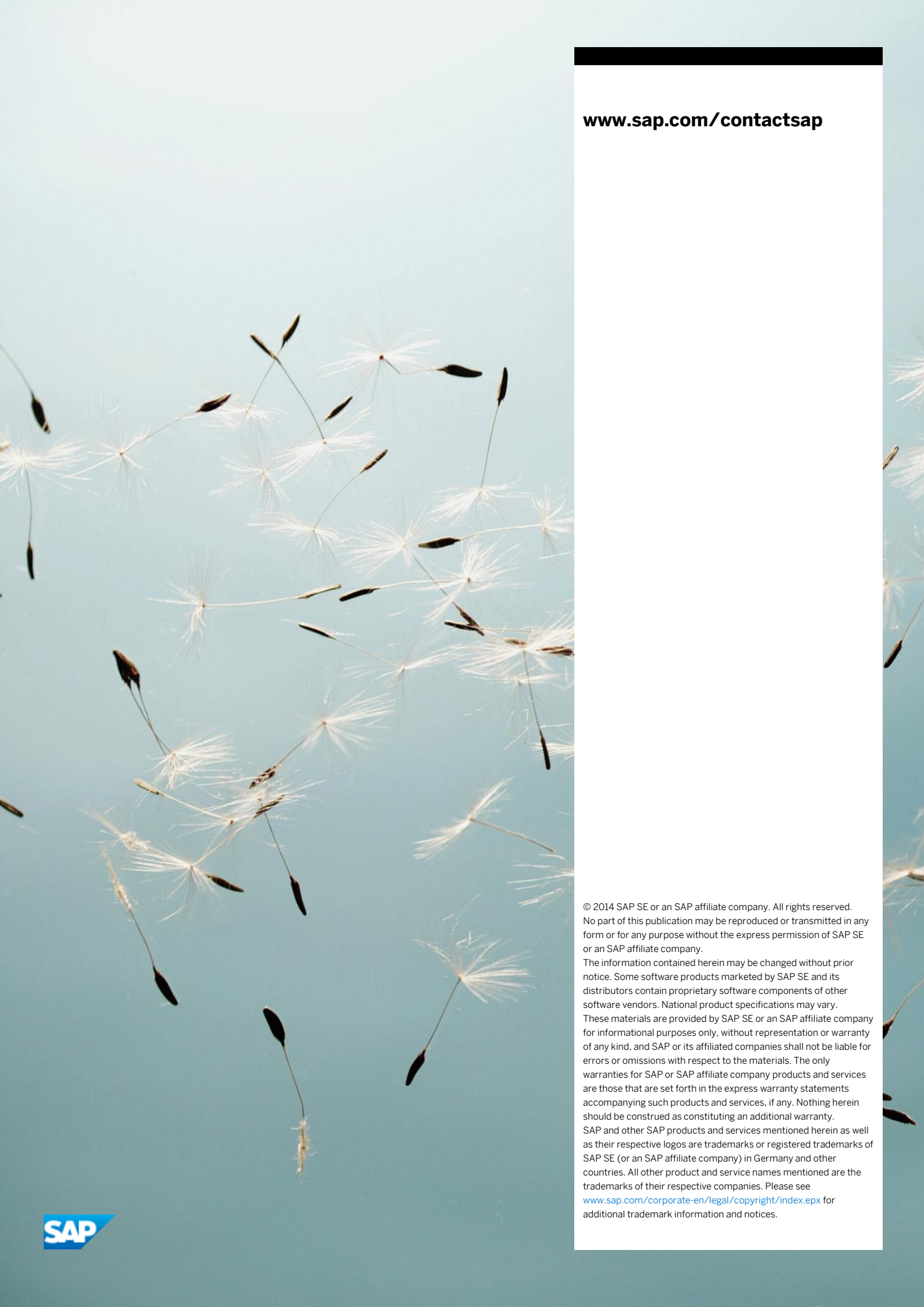
Constants should be defined as data type String.

Data type string can basically be used for all kind of data.

6.3.8 Field Mapping between External System and EHS Management

1:1	Use the standard type STRING.
N:1	When creating the SIT table, multiple field contents have to be merged into a single SIT column (SQL). This is often a case for the type NUMWITHUNIT.
1:N	Type lists may help in this case. Like “STRING.INTERM<“.“>;NUMBER“ could split values like „ca. 5“ into „ca.“ and „5“. A more powerful but also more complex alternative are regular expressions.
(N→1):1	This means that field values from the external system shall go into different EHS data elements depending on the content. There are several possibilities: Use TYPEDEF Use alternative targets and selection lists Example for density: Target: M:[SAP_EHS_1013_005_VALUE,SAP_EHS_1013_005_VALUE_NS] Type def.: NUMBERWITHUNIT["g/cm³",“g/ccm“] Values with unit "g/cm³“ or "g/ccm“ are written to SAP_EHS_1013_005_VALUE. All other values are written to SAP_EHS_1013_005_VALUE_NS. The functions CMP and CMPNC enhance the scope of alternative targets. Alternatives can be chosen depending on the value of a separate column.

Table 61: Field mapping between external system and EHS Management



www.sap.com/contactsap