Cerence Inc.

Categorized Orthographic Phonetic

(COP) File Format

File Format Specification

Document Status:	Released
Document Number:	DOC-41211-055
Version:	01-03
File name:	FileFormat_COP.doc
Date:	12.02.2020
Pages:	19
Author:	V. Zhykharevych

FileFormat_COP.doc

V01-03, Feb-20

Table of Contents

1. Purpose of this document

The COP file format is specified here. It is based on the ECO file format and the ECO Application Framework as defined in [1]. This offers simple database functionality. In the COP format, arbitrary numeric ID columns and columns containing orthographies and phonetic transcriptions are possible. Orthography and transcription are linked via a category name.

1.1. Syntax specification

Syntax is given by modified EBNF notation:

- Non-terminals are always in angle brackets '<', '>'
- Terminals are always in quotes
- Spaces are allowed in the specification string. However, a space has no meaning and does not specify a space in the syntax. <Space> (standing for a single blank space) or <Whitespace> (defined in [1]) have to be used.
- Line breaks are never specified
- '(', ')' are used for grouping
- postfix '*' means 0, 1 or many concatenated instances
- postfix '+' means 1 or many concatenated instances
- postfix '?' means optional instance
- infix '|' denotes alternatives

2. COP

2.1. Basics

The COP file format offers a lightweight database functionality, where Entries can be added to a table, removed from it, and extended by additional values. An Entry can contain arbitrary ID information and arbitrary categorized orthographic and phonetic descriptions.

The COP format makes use of the ECO file format and the ECO Application Framework (ECO-AF), both specified in [1]. The COP format is actually a family of the file formats COP1 and COP2, where COP2 offers more functionality than COP1.

2.2. Columns and parameters

The COP1 and COP2 specifications are given simultaneously.

ECO Scheme

The COP1 specification is referenced in a file by setting ECOScheme to "COP1 <ECOSchemeVersion>".
The COP2 specification is referenced in a file by setting ECOScheme to "COP2 <ECOSchemeVersion>".

The name is built from ASCII letters and underscore. The version string has the format:

```
"V" <DecimalDigit>+ "." <DecimalDigit>+
```

Columns

A COP file can contain arbitrary columns. However, only columns with names ending in "_ID", "_Ortho" and "_Pho" can be used as data columns for entry enrollments in speech recognition applications, holding identifiers, orthographies and phonetic transcriptions, respectively.

The next subsections list special columns and their properties.

Columns x_ID

An *x_ID* column is a column whose name has the format

```
<CategoryName> "_ID"
```

where <CategoryName> is an arbitrary string consisting of ASCII characters.

An x_ID column must be a *numeric ID column* in the ECO-AF sense (see [1]) holding entries given in the ECO *Ident* format.

Each entry must either be Undefined or a single identifier. In AddEntryMode, an identifier is mandatory.

For an x_ID column the parameters NumIDFormat and NumIDBinLength have to be specified. The value of parameter CardinEDBUpdate, if used, must be 0, 1 or 0..1; for AddEntryMode, the value must be 1.

Columns x_Ortho

An *x_Ortho* column is a column whose name has the format

```
<CategoryName> "_Ortho"
```

where <CategoryName> is an arbitrary string consisting of ASCII characters.

Entries of an x_Ortho column must be given in the ECO *Quoted* format. Each entry must be either Undefined or a single string.

The value of parameter CardinEDBUpdate, if used, must be 0, 1 or 0..1.

The value of the parameter *SplitCharSequence*, if used, can be any string, surrounded by double quotes. Escaping works like in the rest of the COP file. By default, this value is not set, meaning no splitting is applied.

Columns x_Pho

An *x_Pho* column is a column whose name has the format

```
<CategoryName> "_Pho"
```

where <CategoryName> is an arbitrary string consisting of ASCII characters.

An x_Pho column must be a phonetic column in the ECO-AF sense (see [1]) with entries given ECO *Quoted* format.

Each entry must either be Undefined or a single string.

For an x_Pho column the parameters PhoneticAlphabet, LangIdScheme and PhoneticLangSpecs have to be specified. Their values must be the same in all x_Pho columns. The value of parameter CardinEDBUpdate, if used, must be 0, 1 or 0..1

A column entry models a list of space separated phonetic transcriptions within a single string. This means, the target string has the following format:

```
<PhoneticTranscription>( <Space>+ <PhoneticTranscription>)*
```

Here, <PhoneticTranscription> is a phonetic transcription alternative and may not contain any whitespace character. The phonetic transcription is given in the phonetic alphabet that is specified by the PhoneticAlphabet parameter.

Not all phonetic alphabets are supported in each system. It must be clarified with Cerence which phonetic alphabets are supported. Details about the phonetic description must always be clarified with Cerence.

Example

#ECO V1.2 UTF-8;
#ECOScheme COP1 V1.6
#EDBDatabaseID DiBa
#EDBTableID Keywords

#Column1 Keyword_ID
#Column2 Keyword_Ortho
#Column3 Keyword_Pho

#Format(Keyword_ID) Ident
#AFType(Keyword_ID) NumID
#NumIDFormat(Keyword_ID) Decimal
#NumIDBinLength(Keyword_ID) 32

Keyword_ID: Unique ID value
Keyword_Ortho: Value of XML "keyword" attribute.
Keyword_Phon: Value of XML "mainphonem" attribute. If multiple transcriptions, separate with whitespace.

1;"Reifendruck";"#Ra&i.fEn_'tRUk# #'Ra&i.fEn_'tRUk_'pRy:.f$n_ma.nU.'El#"

Column EDBTableUpdateMode

A COP file is an ECO DB file that can chose between two file level update modes, AddEntryMode and mixed update mode. If a COP file uses mixed update modes, the column EDBTableUpdateMode is mandatory. It must be in the Ident format and a column entry must be one of the identifiers

- a, r in COP1
- a, r, e, c in COP2

The value of parameter Cardin, if used, must be 1. The parameter CardinEDBUpdate is prohibited.

Undefined values

Undefined (e.g. unknown) values must be written as blank column entries (i.e. empty columns), not as empty or blank strings within quotes (like "" or " ").

Consistency of phonetic transcriptions

The phonetic column parameters must have identical values in all phonetic columns, i.e. there must be a common phonetic alphabet, a common language identification scheme, and a common set of phonetic language specifications.

2.3. Consistency of COP files

All COP files updating the same table must have consistent column structure. This means,

- values of parameter EDBTableUpdateMode must be identical,
- column names must be identical and in the same order, and
- ECO-AF column properties (e.g. given by column parameters) must be compatible in the files.

In particular,

- numeric ID specifications (NumIDFormat and NumIDBinLength) must have the reference to the same column name;
  - #NumIDFormat(Entry_ID) BSBase64
  - #NumIDBinLength(Entry_ID) 64
- phonetic specifications (PhoneticAlphabet and LangIdScheme) must have the reference to the same column name;
  - #PhoneticAlphabet(Entry_Pho) LH_Plus
  - #LangIdScheme(Entry_Pho) StarRec_llCC
- the mapping tables PhoneticLanguage→Version (specified by the PhoneticLangSpecs parameter) must be non-contradictory;
- allowed values (given by the Values parameter) must be identical and in the same order;

2.4. Database functionality

The database functionality is the one from ECO-AF with some restrictions.

Update mode specification on file level

A COP file is an ECO DB file that can chose between two update mode specifications, given by the parameter EDBTableUpdateMode:

EDBTableUpdateMode	Update mode of line
AddEntryMode	AddEntryMode
GivenInColumn	depending on the entry of column EDBTableUpdateMode

A file of the second type is called a *mixed update mode file*.

Update modes

In a mixed update mode file, the update mode of a single line is determined by the entry of column EDBTableUpdateMode, which can have the values

- a, r in COP1.
- a, r, e, c in COP2.

The meaning of the identifiers is

Ident.	Update mode of line
a	AddEntryMode
r	RemoveEntryMode
e	ExtendEntryMode
c	RemoveContainsEntryMode

Selecting entries and applying common operations

A line with RemoveEntryMode, RemoveContainsEntryMode or ExtendEntryMode is used

- to specify how to select certain entries from the table L via filter conditions and
- to specify a common operation to all selected entries (remove entries or extend them by common data).

The key columns used to select entries may only consist of x_ID columns, and in each line at most one key column may have a defined value (it is possible that all key columns have Undefined values which selects all entries). It is legitimate that some x_ID columns are value columns and not key columns.

This works as follows:

- Arbitrary x_ID columns can be stipulated (and possibly tagged) as filter columns, e.g. columns C_1, …, C_m.
- In each RemoveEntryMode or RemoveContainsEntryMode line, one of these filter columns must have a defined value (a single value, not a list).
- In each ExtendEntryMode line, one or none of these filter columns may have a defined value (a single value, not a list).
- A line where filter column C_k contains a defined value x_k triggers the following actions:
  - select all entries in the current table for which $C_k=x_k$ holds
  - apply a common operation to each of the entries selected by the filtering operation
- A line where all filter columns have Undefined value triggers the following action:
  - apply a common operation to all entries of the table

The key columns can formally be given by the EDBKeyColumns parameter family.

AddEntryMode

The optional EDBKeyColumns(AddEntryMode) parameter can be given to express that a certain x_ID column (or column set) is a primary key to the EDB Entries (which is not necessarily the same as the entries of an external database), asserting that for each new EDB Entry a new identifier is provided.

RemoveEntryMode/ RemoveContainsEntryMode

All x_ID columns are allowed as key columns, but in each line at most one column may have a defined value. The optional EDBKeyColumns(RemoveEntryMode) parameter can be given to list these columns formally.

The diffrerence between ‘RemoveEntryMode’ and ‘RemoveContainsEntryMode’ consist in the removal match type. With ‘RemoveEntryMode’, entries of which every terminal matches the query will be removed. With ‘RemoveContainsEntryMode’, entries that contain the terminal sequence specified in the query will be removed.

ExtendEntryMode

This mode is supported in COP2, but not in COP1.

All x_ID columns are allowed as key columns, but in each line at most one key column may have a defined value (it is possible that all key columns have Undefined values which selects all entries). The EDBKeyColumns(ExtendEntryMode) parameter or the EDBValueColumns(ExtendEntryMode) parameter must be given in order to specify which values are used for entry selection and which values are seen as content.

To specify which extension operation is actually performed on a value column, for each column an optional parameter COPExtendMode(<ColumnName>) can be given. The allowed values are:

- *AppendItems*: The new list is joined to the current list, i.e. the values from the new list are appended to the current list.
- *AppendNewItems*: Similar to AppendItems, but only new values are appended to the current list.
- *AppendList*: The ECO table column contains lists of string lists. The new string list is appended to the list of string lists.

The default operation is *AppendItems*.

Note: *The entry of an x_Pho column is interpreted as list of transcriptions here (represented as space separated substrings within one string) and not as single string.*

The AppendList mode is needed when the lists of different columns have to be synchronized.

Example:

Our table has the columns "Entry_ID", "Artist_Ortho" and "Artist_Pho" and contains names of artists. Suppose that for an artist several names are possible, like for Elvis who is also referred to as "The King". Each name may have several pronunciation alternatives. Suppose that Elvis has the pronunciation alternatives "#'ElvIs#" and "#'Elv$s#".

Using the AppendList mode for column Artist_Pho, the resulting lists of string lists are:

Artist_Ortho:	(	"Elvis",	"The King"	)
Artist_Pho:	(	(""#'ElvIs#", "#'Elv$s#"),	("#T$_'kInK#")	)

Since we have a list of string lists in the second row, a correspondence between orthography and transcription alternatives is possible. If we had used the AppendItems mode, the second row would contain a flat list of 3 strings; here, no correspondence would be possible.

Update type of a file

A table (identified by an EDBTableID) is seen as a container and can either be

- updated incrementally, or
- cleared and filled anew

by a table update file. The optional parameter EDBTableUpdateType describes for which kind of operation the ECO DB file is used. The allowed values are:

- *FullUpdate*: The old table content is erased and new content is given by the file.
- *IncrementalUpdate*: The old table content is modified by the update operations of the file.

The default FullUpdate is used in case of a missing parameter.

Note that a FullUpdate does not define a new table (with new EDBTableID), but rather changes the content of the table, i.e. the EDBTableID stays the same.

Parameters for ECO DB file handling

The parameters EDBTableUpdateMode and EDBDatabaseID are mandatory. All other database ID-parameters and parameter EDBTableUpdateType are optional. However, applications may mandate their usage.

The value of each database ID-parameter is an arbitrary string. It is only used to make linkages between files and is not interpreted in any way.

The update mode dependent EDBKeyColumns parameters can be given in order to specify key columns. For ExtendEntryMode one of the parameters EDBKeyColumns and EDBValueColumns must be given.

For specifying the cardinality of a column, the mode dependent CardinEDBUpdate parameters can be used. The parameter Cardin should not be used (except for column EDBTableUpdateMode) since in a mixed update mode file the use of this parameter is subtle and could lead to misunderstandings.

Software restrictions

Please note that this is only a modeling concept and not a specification of a specific software implementation. To manage a database and execute operations modeled by ECO DB files, an appropriate ECO DB manager must be implemented. It must be clarified with Cerence what subset of the DB functionality is needed.

2.5. Example: Address Book

A simple address book example is given using mixed update modes. The various use cases are illustrated in the following subsections, all acting on the same ECO DB table.

The basic file header

All update files in this example make use of the following header, except that EDBTableUpdateID must be different in every update file. Possible modifications are given in the next subsection.

```
#ECO V1.2 UTF-8;
#ECOScheme COP1 V1.2

#EDBDatabaseID AddressBook
#EDBTableID AddressBook_UserDad
#EDBTableUpdateMode GivenInColumn
#EDBTableUpdateType IncrementalUpdate
# The following ID has to be different in every update file:
#EDBTableUpdateID AddressBook_UserDad_2010-10-01_16-09-34

#Column1 EDBTableUpdateMode
#Column2 Entry_ID
#Column3 ContactType_ID
#Column4 LastName_Ortho
#Column5 LastName_Pho
#Column6 FirstName_Ortho
#Column7 FirstName_Pho
#Column8 PhoneNumber_Ortho

# ===EDBTableUpdateMode===
# The update mode must always be given.
#Cardin(EDBTableUpdateMode) 1
#Format(EDBTableUpdateMode) Ident

# ===Keys===
#EDBKeyColumns(AddEntryMode) Entry_ID
#EDBKeyColumns(RemoveEntryMode) Entry_ID, ContactType_ID

# ===Content columns===

# ---Entry_ID---
# Entry_ID is the primary key and therefore mandatory for new entries.
#CardinEDBUpdate(AddEntryMode)(Entry_ID) 1
#Format(Entry_ID) Ident
#AFType(Entry_ID) NumID
#NumIDFormat(Entry_ID) Decimal
#NumIDBinLength(Entry_ID) 32
```

```
# ---ContactType_ID---
# The contact type is a mandatory identifier and must be selected from
# a list of defined values.
#CardinEDBUpdate(AddEntryMode)(ContactType_ID) 1
#Format(ContactType_ID) Ident
#AFType(ContactType_ID) NumID
#NumIDFormat(ContactType_ID) Decimal
#NumIDBinLength(ContactType_ID) 8
#Values(ContactType_ID) 0, 1, 2
# 0=Friend/Acquaintance
# 1=Business
# 2=Restaurant/Venue

# ---LastName_Ortho---
# This is a mandatory orthography.
#CardinEDBUpdate(AddEntryMode)(LastName_Ortho) 1

# ---LastName_Pho---
# This is an optional transcription.
#CardinEDBUpdate(AddEntryMode)(LastName_Pho) 0..1
#AFType(LastName_Pho) Phonetic
#PhoneticAlphabet(LastName_Pho) LH_Plus
#PhoneticLangSpecs(LastName_Pho) ENU 0.0
#LangIdScheme(LastName_Pho) Nuance_LLC

# ---FirstName_Ortho---
# This is an optional orthography.
#CardinEDBUpdate(AddEntryMode)(FirstName_Ortho) 0..1
#SplitCharSequence(FirstName_Ortho) "\\;"

# ---FirstName_Pho---
# This is an optional transcription.
#CardinEDBUpdate(AddEntryMode)(FirstName_Pho) 0..1
#AFType(FirsttName_Pho) Phonetic
#PhoneticAlphabet(FirstName_Pho) LH_Plus
#PhoneticLangSpecs(FirstName_Pho) ENU 0.0
#LangIdScheme(FirstName_Pho) Nuance_LLC

# ---PhoneNumber_Ortho---
# This is an optional phone number.
#CardinEDBUpdate(AddEntryMode)(PhoneNumber_Ortho) 0..1
```

Variations of the file header

Different EDBTableUpdateID values

In each update file a different EDBTableUpdateID has to be used.

Variation: Omitting optional parameters

Some parameters can be omitted when consistency checks are not desired. These are:

- `Cardin`
- `CardinEDBUpdate`
- `Values`
- `AFType`
- `EDBTableID`
- `EDBTableUpdateID`

Whether a parameter can be omitted depends on the application and must be clarified with Cerence.

Variation: Update type

If the DB table is empty or should be cleared before adding new entries, the header must contain the line

```
#EDBTableUpdateType FullUpdate
```

instead of

```
#EDBTableUpdateType IncrementalUpdate
```

Variation: Additional dynamic table ID parameter

In addition, the identifying parameter EDBPreviousTableUpdateID can be used to describe the update sequence with a linked list. Its usage depends on the application.

The file body

Adding Entries

This file body shows how to add Entries to the DB table.

```
a; 0; 1; "Burns"; "#'bE0R+nz#"; "Montgomery"; "#mAnt.'g^m.R+i#"; "555-0113"
a; 1; 0; "Carlson"; "#'kAR+l.s$n# "; "Carl"; "#'kAR+l#"; "555-1867"
a; 2; 2; "Moe's Tavern"; "#'mo&Uz_'t@.v$R+n#"; ; ; "555-1239"
a; 3; 1; "Smithers"; "#'smI.T$R+z#"; "Waylon"; "#'we&I.l$n#"; "555-5246"
a; 4; 0; "Simpson"; ; "Abe"; ; "555-7878"
a; 5; 0; "Grimes"; ; "Frank"; ; "555-0815"
```

Removing a single Entry

This line shows how to remove a single Entry from the table using its primary key.

```
# Remove Frank Grimes, i.e. Entry with Entry_ID 5.
r; 5; ; ; ; ; ;
```

Removing a single partial Entry

This line shows how to remove a single Entry from the table using only partial orthography.

```
# Remove Moe’s Tavern.
c; ; ; "Tavern"; ; ; ;
```

Removing several Entries at once

This line shows that several Entries can be removed at once using the selection mechanism with a key different from the primary key. Here the key ContactType_ID is used to select entries.

```
# Remove all Entries associated with ContactType "Business". This will result
# in removing Burns and Smithers.
r; ; 1; ; ; ; ;
```

Replacing the content of an Entry

This file body shows how to replace some Entry's content by first removing and then re-adding the Entry with updated content.

```
# Moe's Tavern has a new ISDN phone number. Replace the phone number by
# removing and re-adding the table entry.
r; 2; ; ; ; ;
a; 2; 2; "Moe's Tavern"; "#'mo&Uz_'t@.v$R+n#"; ; ; "555-9311"
```

Extending the content of an Entry (COP2 only)

If the content of an entry changes in a way that information is added to a list, this can be done by using the ExtendEntryMode.

```
# Add a less official name for "Abe Simpson".
e; 4; ; "Dad"; "#'d@d#"; ;
# Here, a new orthography and a new transcription is added. Orthographies
# and transcriptions are kept in separate lists. (Whether the lists
# are synchronized or not depends on the parameter COPExtendMode.)
```

This only works in COP2 and the header must acknowledge this with the ECOScheme parameter:

```
#ECOScheme COP2 V1.2
```

Moreover, in order to specify that the column used to select entries is Entry_ID (and that the contents of all other table columns are considered additional values), the header must include the following line:

```
#EDBKeyColumns(ExtendEntryMode) Entry_ID
```

A Appendix

A.1 Glossary and Definitions

Term	Definition

A.2 Related References

ID	Document Name	Link to Document

A.3 Version History

Version	Description of Change	Author	Date
1.0	Takeover to Nuance	N. Does	01.10.2010
1.01	Added note on SplitCharSequence	O. vanPorten	20.02.2014
1.02	Added RemoveContainsEntryMode	S. Lahlou Mimi	02.07.2015
1.03	Adapted to Cerence layout	J. Van Looveren	12.02.2020

A.4 Authorization

Document Owner	
Name	Role
V. Zhykharevych	

Approved by			
Name	Role	Approval	Date
Luc van Tichelen	Sr. Director Automotive PS		
Roy Forsberg	Sr. Director Mobile Program & Quality Management		

Reviewed by			
Name	Role	Reviewed	Date