Annex G
(normative)
Generating longform EXPRESS schemas from shortform schemas

G.1 Introduction

This edition of ISO 10303-11 adds functionality for creating data models that consist of several dependent schemas, also called modules. An incomplete data specification may be defined in one SCHEMA and may be reused by another SCHEMA to be possibly specialized and extended. A high level SCHEMA may collect several such resources into one top level SCHEMA to build a complete data specification; this high level SCHEMA may be referred to as the shortform of an application schema. The result of building the complete data specification is called the longform of the application schema.

NOTE The terms shortform and longform are defined in ISO 10303-1:2002.

As long as the implementation methods included in ISO 10303 are not updated to this edition of ISO 10303-11 or implementations are not updated to work with the new implementation methods, it may be useful to convert a data specification that is in accordance to ISO 10303-11:2003 to a data specification that is in accordance to ISO 10303-11:1994. This annex specifies the rules for how such a conversion shall be done. Especially, the rules enable the conversion of a ISO 10303-11:2003 shortform schema to a ISO 10303-11:1994 longform schema. These rules are designed to result in a complete and consistent longform and to minimize the loss of semantics of the original shortform data model.

NOTE Conformance requirements to this specification are stated in clause 4 and annex E.

G.2 Fundamental concepts

Input to the longform generation process is a shortform EXPRESS SCHEMA with USE and REFERENCE statements, and one or more additional SCHEMAṪhe process results in a new SCHEMA that contains all constructs defined in the shortform, adds the constructs that are explicitly interfaced by USE and REFERENCE, and resolves all references to other constructs according to clause 11 of this document.

The resulting longform is an almost semantically identical model without the USE and REFERENCE statements; referenced objects are brought directly into the schema. Information describing the originating SCHEMA and how objects were interfaced (implicitly or explicitly and by USE or REFERENCE) is discarded. Objects in the interfaced schemas that are not directly referenced by the objects in the shortform are discarded. Pruning and rewriting of some of the constructs of the input schemas takes place to omit the objects that are originally declared, but are not used in the final schema.

Specifically, the longform generation has the following impact whether or not this involves the conversion from an ISO 10303-11:1994 schema to an ISO 10303-11:2003 schema:

— Loss of knowledge of the schema that each construct originates from.

— Loss of knowledge of how the constructs were interfaced, i.e., their visibility and instantiability are not known. Information describing how the object was interfaced (distinction between USE and REFERENCE), affects its instantiability and visibility; see clause 11 paragraphs 2 and 3. These implications are lost in the resulting longform schema.

— Selectable items of a SELECT type are pruned to remove those items that are not interfaced into the schema. According to 11.4.2, selectable items are not implicitly interfaced as a result of the interfacing of the SELECT type itself. If the selectable items were left in the select list, and the objects were not visible in the schema, the compilation of the longform schema would result in errors.

— SUBTYPE and SUPERTYPE statements in entities are pruned to reflect the pruning of the subtype/supertype graph as described in 11.4.3 and annex C.

— Remarks may be discarded.

— The case of user identifiers may not be preserved.

— SCHEMA names in fully qualified attribute references are replaced with the SCHEMA name of the longform.

The conversion from an ISO 10303-11:2003 shortform to an ISO 10303-11:1994 longform requires the removal or change of EXPRESS constructs that are not supported by ISO 10303-11:1994. In particular the following actions shall be performed:

a) The schema version identifier is converted into a remark.

b) EXTENSIBLE SELECT data types are resolved into SELECT data types according to ISO 10303-11:1994, which are not extensible.

c) EXTENSIBLE ENUMERATION data types are resolved into ENUMERATION data types according to ISO 10303-11:1994, which are not extensible.

d) SUBTYPE_CONSTRAINT statements are removed:

   1) their SUPERTYPE constraints and ABSTRACT statements are converted into SUPERTYPE statements according to ISO 10303-11:1994 and are rewritten to remove types that would otherwise not appear in the longform schema;

   2) TOTAL_OVER constraints are replaced by RULE constructs.

e) ABSTRACT entities and GENERIC_ENTITY are converted into ABSTRACT SUPERTYPE constraints.

f) RENAMED constructs are converted into DERIVE attributes.

g) Errors shall be produced for empty SELECT data types and for RULE headers with no parameters.

The following sub-clauses describe these actions in more detail.

G.3 Conversion of ISO 10303-11:2003 constructs to ISO 10303-11:1994 constructs

G.3.1 Conversion of schema version identifier

The STRING that represents a schema version identifier (see 9.3) shall be enclosed within an embedded remark that is located right after the schema name separated by a blank and before the semicolon that completes the SCHEMA declaration. The enclosing quotes for the STRING shall be included in the remark. The STRING shall be preceded by the STRING `'schema_version_id = '` without the quotes that are used here to distinguish the string from the rest of this sentence.

EXAMPLE 1 The following ISO 10303-11:2003 schema:

```
SCHEMA geometry_schema 'version_1';
END_SCHEMA;
```

is converted into the following ISO 10303-11:1994 schema:

```
SCHEMA geometry_schema (* schema_version_id = 'version_1' *);
END_SCHEMA;
```

G.3.2 Conversion of extensible constructed data types

For the conversion of extensible select data types and extensible enumeration data types (see 8.4), a schema shall be chosen as the context schema for the conversion. The completion of the extensions shall be evaluated within the context schema. For each extensible constructed data type that is visible in the

context schema and that is not defined in the context schema, it is necessary to define the completed enumeration or select list in the schema that the type is defined in. The result of this process is that a context schema-specific schema is created for each schema that is visible in the context schema.

EXAMPLE 1 If the context schema was the shortform AIM of an Application Protocol and if it and the integrated resources used by this shortform contained extensible constructed data types, the result of this process would be a set of supporting schemas that were derived from the interfaced integrated resource schemas and that could only be used in the context of this AP.

For each extensible or extending data type in any schema that is visible to the context schema, the following conversions shall be performed:

— each extensible or extending data type shall be replaced by a data type of the same name that is neither extensible nor an extension of any data type;

— the list items specified in each constructed data type shall be the set of items within the domain of the data type as evaluated with respect to the context schema;

— in the case that a defined data type in the schema that is visible in the context schema has an extensible or extending data type as its underlying data type, add a WHERE clause to that defined data type; this shall eliminate all items that result from the conversion and that are not valid list items in the set of schemas that are visible in the context schema;

— to maintain the dependencies among the extensible and extending data types their target constructs shall be related; depending data types, i.e., the extending ones, shall be created as defined data types that use the extensible data type as underlying type; the elements of the underlying data type that are not valid in the defined data type shall be constrained using local rules; see below for examples.

The following sub-clauses give some details on how to implement these rules for EXTENSIBLE ENUMERATION and EXTENSIBLE SELECT data types.

G.3.2.1 Extensible enumeration

EXTENSIBLE ENUMERATION (see 8.4.1) shall be converted according to the procedure above to enumeration data types that are not extensible. The name of the source construct shall be maintained in the target.

An enumeration data type that is based on an extensible enumeration data type shall be converted to a defined data type that has as underlying data type the target enumeration data type of the extensible enumeration that it is based on. The name of the defined data type in the target model shall be the name of the corresponding enumeration data type in the source. The defined data type shall exclude enumeration items that are invalid in its context, but that are specified for its underlying enumeration, from instantiation in the context schema by local rules.

EXAMPLE 1 Source schema according to ISO 10303-11:2003:

```
SCHEMA S1;

TYPE general_approval = EXTENSIBLE ENUMERATION OF (approved, rejected);
END_TYPE;

END_SCHEMA;
```

Target in ISO 10303-11:1994 longform for `S1` as context schema:

```
TYPE general_approval = ENUMERATION OF (approved, rejected);
END_TYPE;
```

Source schema according to ISO 10303-11:2003 :


```
SCHEMA S2;

USE FROM S1 (general_approval);

TYPE domain2_approval = EXTENSIBLE ENUMERATION BASED_ON general_approval WITH
(pending);
END_TYPE;

END_SCHEMA;
```

Target in ISO 10303-11:1994 longform for S2 as context schema:

```
TYPE general_approval = ENUMERATION OF (approved, rejected, pending);
END_TYPE;

TYPE domain2_approval = general_approval;
END_TYPE;
```

Source schema according to ISO 10303-11:2003 :

```
SCHEMA S3;

USE FROM S1 (general_approval);

TYPE domain3_approval = EXTENSIBLE ENUMERATION BASED_ON general_approval WITH
(cancelled);
END_TYPE;

END_SCHEMA;
```

Target in ISO 10303-11:1994 longform for S3 as context schema:

```
TYPE general_approval = ENUMERATION OF (approved, rejected, cancelled);
END_TYPE;

TYPE domain3_approval = general_approval;
END_TYPE;
```

Source schema according to ISO 10303-11:2003 :

```
SCHEMA S4;

USE FROM S2 (domain2_approval);

REFERENCE FROM S3 (domain3_approval);

TYPE specific_approval = ENUMERATION BASED_ON domain2_approval WITH (rework);
END_TYPE;

END_SCHEMA;
```

Target in ISO 10303-11:1994 longform for S4 as context schema:

```
TYPE general_approval = ENUMERATION OF (approved, rejected, pending,
cancelled, rework);
END_TYPE;

TYPE domain2_approval = general_approval;
WHERE
```



```
  WR1: SELF <> cancelled;
END_TYPE;

CONTINUE

TYPE domain3_approval = general_approval;
WHERE
  WR1: SELF <> pending;
  WR2: SELF <> rework;
END_TYPE;

TYPE specific_approval = general_approval;
WHERE
  WR1: SELF <> cancelled;
END_TYPE;
```

EXAMPLE 2 Source schema according to ISO 10303-11:2003 :

```
SCHEMA EXPORT;

TYPE colour = EXTENSIBLE ENUMERATION;
END_TYPE;

TYPE stop_light = ENUMERATION BASED_ON colour WITH (red, yellow, green);
END_TYPE;

END_SCHEMA;

SCHEMA IMPORT;

USE FROM EXPORT (stop_light);

TYPE canadian_flag = ENUMERATION BASED_ON colour WITH (red, white);
END_TYPE;

END_SCHEMA;
```

Target enumerations in ISO 10303-11:1994 longform for `IMPORT` as context schema:

```
TYPE colour = ENUMERATION OF (red, white, yellow, green);
END_TYPE;

TYPE canadian_flag = colour;
END_TYPE;

TYPE stop_light = colour;
WHERE
  WR1: SELF <> white;
END_TYPE;
```

Special attention needs to be given to the fact that enumeration data types according to ISO 10303-11:1994 of this document imply an order of the enumeration items. This concept of order is not present in ISO 10303-11:2003 .

G.3.2.2 Extensible select

Extensible selects (see 8.4.2) shall be converted according to the procedure above to select data types that are not extensible. The name of the source construct shall be maintained in the target.

A select data type that is based on an extensible select data type shall be converted to a defined data type that has as underlying data type the target select data type of the extensible select that it is based on. The name of the defined data type in the target model shall be the name of the corresponding select data type in the source. The defined data type shall exclude select items that are invalid in its context, but that are specified for its underlying select, from instantiation in the context schema by local rules.

EXAMPLE 1 Source schema according to ISO 10303-11:2003 :

```
SCHEMA EXPORT;

TYPE attachment_method = EXTENSIBLE SELECT(nail, screw);
END_TYPE;

ENTITY nail;
END_ENTITY;

ENTITY screw;
END_ENTITY;

END_SCHEMA;

SCHEMA IMPORT;

USE FROM EXPORT (attachment_method);

TYPE permanent_attachment = SELECT BASED ON attachment_method WITH (glue, weld);
END_TYPE;

TYPE simple_attachment = SELECT BASED ON attachment_method WITH (needle, tape);
END_TYPE;

...;

END_SCHEMA;
```

Target selects in ISO 10303-11:1994 longform for `IMPORT` as context schema:

```
TYPE attachment_method = SELECT (nail, screw, glue, weld, needle, tape);
END_TYPE;

TYPE permanent_attachment = attachment_method;
WHERE
  WR1: NOT (('IMPORT.NEEDLE') IN TYPEOF(SELF));
  WR2: NOT (('IMPORT.TAPE') IN TYPEOF(SELF));
END_TYPE;

TYPE simple_attachment = attachment_method;
WHERE
  WR1: NOT (('IMPORT.GLUE') IN TYPEOF(SELF));
  WR2: NOT (('IMPORT.WELD') IN TYPEOF(SELF));
END_TYPE;
```

EXAMPLE 2 Source schema according to ISO 10303-11:2003 :

```
SCHEMA SELECT_TREE_EXPORT;

TYPE t1 = EXTENSIBLE SELECT (tata);
END_TYPE;
```



```
TYPE t2 = EXTENSIBLE SELECT BASED_ON t1 WITH (titi);
END_TYPE;

TYPE t3 = EXTENSIBLE SELECT BASED_ON t2;
END_TYPE;

ENTITY tata;
END_ENTITY;

ENTITY titi;
END_ENTITY;

END_SCHEMA;

SCHEMA SELECT_TREE_IMPORT;

USE FROM EXPORT (t1, tata, t2, titi, t3);

TYPE t4 = SELECT BASED_ON t1 WITH (toto);
END_TYPE;

ENTITY toto;
END_ENTITY;

END_SCHEMA;
```

Target selects in ISO 10303-11:1994 longform for `SELECT_TREE_IMPORT` as context schema:

```
TYPE t1 = SELECT (tata, titi, toto);
END_TYPE;

TYPE t2 = t1;
WHERE
  WR1: NOT (('SELECT_TREE_IMPORT.TOTO') IN TYPEOF(SELF));
END_TYPE;

TYPE t3 = t2;
END_TYPE;

TYPE t4 = t1;
WHERE
  WR1: NOT (('SELECT_TREE_IMPORT.TITI') IN TYPEOF(SELF));
END_TYPE;
```

G.3.3 Conversion of subtype constraints

The keyword SUBTYPE_CONSTRAINT is not a part of ISO 10303-11:1994 and shall, therefore, be eliminated by the conversion process. The semantics of the constraint, however, shall be maintained in the ISO 10303-11:1994 longform.

The following conversion rules apply to the SUBTYPE_CONSTRAINT in general:

— The constraints on the subtype/supertype graph in the context of the context schema shall be added to all the schemas that are visible to the context schema.

— The SUBTYPE_CONSTRAINT is deleted from all schemas that are visible to the context schema and replaced by semantically equivalent constructs.

The conversions of TOTAL_OVER constraints and constraints on valid instantiations of subtype/supertype graphs are described in the following sub-clauses.

G.3.3.1 Total over constraint

The following conversion rules apply to TOTAL_OVER constraints:

— The longform schema shall maintain the semantics of the TOTAL_OVER constraint even if only one of the constituents of the constraint is interfaced.

— For each TOTAL_OVER constraint of a SUBTYPE_CONSTRAINT a global RULE shall be added to the SCHEMA.

— The RULE name shall be `total_over_<subtype constraint name>`.

— The RULE shall be valid FOR the supertype entity that the TOTAL_OVER was specified for.

— The WHERE rule in the global RULE shall ensure that each instance of the target supertype is of the data type(s) of the one or several subtypes that are specified in the source TOTAL_OVER constraint and that are interfaced into the target longform schema.

EXAMPLE 1 The example describes a source schema that includes a TOTAL_OVER constraint according to ISO 10303-11:2003 :

```
SCHEMA S1;

ENTITY E1;
END_ENTITY;

ENTITY E2
  SUBTYPE OF (E1);
END_ENTITY;

ENTITY E3
  SUBTYPE OF (E1);
END_ENTITY;

SUBTYPE_CONSTRAINT a_total_over_example FOR E1;
  ABSTRACT SUPERTYPE;
  TOTAL_OVER (E2, E3);
END_SUBTYPE_CONSTRAINT;

END_SCHEMA;

SCHEMA IMPORT;
  USE FROM S1 (E1, E2);
END_SCHEMA;
```

The longform of schema `IMPORT` maintains the semantics of the TOTAL_OVER constraint even though only one of the constituents of the constraint, `E2`, is used in `IMPORT`, and not `E3`. The semantics of the TOTAL_OVER is in the longform captured by the following global RULE:

```
...

RULE total_over_a_total_over_example FOR (E1);
WHERE
  (* All instances of E1 shall also be of entity data type E2. *)
  WR1: SIZEOF (QUERY(e1_i <* E1 |
       SIZEOF (['IMPORT.E2'] * TYPEOF(e1_i)) = 0)) = 0;
END_RULE;
```


...

G.3.3.2 Subtype/supertype instantiation constraints

If the SUBTYPE_CONSTRAINT includes constraints on which subtype/supertype graphs may be instantiated, these shall be moved into SUPERTYPE constraints. Such constraints typically include the keywords AND, ANDOR, and ONEOF. The SUPERTYPE constraints shall be built by the following rules:

a) For each entity constrained by a SUBTYPE_CONSTRAINT that contains a SUPERTYPE constraint, if the SUPERTYPE constraint only contains entity data types that are defined in the same SCHEMA as the SUPERTYPE, the SUPERTYPE constraint is enclosed in parentheses and added to a SUPERTYPE constraint specified in the entity.

b) Each constraint that results from a SUBTYPE_CONSTRAINT is combined via ANDOR with any other constraints added from different SUBTYPE_CONSTRAINT specifications.

c) If any SUPERTYPE constraint contains entity data types that are not defined in the same longform SCHEMA as the SUPERTYPE, determine all data types that are related by ANDOR by means of the following expression replacement algorithm:

   1) take all SUPERTYPE constraints of the source schema, both those inside and outside of SUBTYPE_CONSTRAINT constructs, and put in all places where data types are referred that are not brought into the longform SCHEMA, empty boxes;

   2) apply for the sub-expressions the following simple rules to make the boxes disappear:

```
ONEOF (A)          => ONEOF (A)
ONEOF (A, B, box) => ONEOF (A, B)
ONEOF (box)        => box
A ANDOR box        => A
A AND box          => error
box AND box        => box
box ANDOR box      => box .
```

EXAMPLE 1 This example is based on `SCHEMA EXAMPLE` in annex B. This schema is in accordance to ISO 10303-11:2003 .

```
SCHEMA EXAMPLE;

ENTITY p;
END_ENTITY;

SUBTYPE_CONSTRAINT p_subs FOR p;
  ONEOF(m, f) AND ONEOF(c, a);
END_SUBTYPE_CONSTRAINT;

ENTITY m SUBTYPE OF (p);
END_ENTITY;

ENTITY f SUBTYPE OF (p);
END_ENTITY;

ENTITY c SUBTYPE OF (p);
END_ENTITY;

ENTITY a ABSTRACT SUBTYPE OF (p);
END_ENTITY;

SUBTYPE_CONSTRAINT no_li FOR a;
```



```
  ONEOF(l, i);
END_SUBTYPE_CONSTRAINT;

ENTITY l SUBTYPE OF (a);
END_ENTITY;

ENTITY i SUBTYPE OF (a);
END_ENTITY;

END_SCHEMA;


SCHEMA IMPORT;

USE FROM EXAMPLE(l);

REFERENCE FROM EXAMPLE(m);

END_SCHEMA;
```

The ISO 10303-11:1994 compliant longform of schema `IMPORT` shall represent the SUBTYPE_CONSTRAINT as SUPERTYPE statements. The following considerations apply:

— `l` causes `a` and `p` to be implicitly interfaced;

— `m` causes `p` to be implicitly interfaced;

— the instantiation of `l` is constrained by the SUBTYPE_CONSTRAINT `no_li` of `a`;

— the instantiations of `m` and `a` are constrained by the SUBTYPE_CONSTRAINT `p_subs` of `p`;

— for both SUBTYPE_CONSTRAINT `p_subs` and SUBTYPE_CONSTRAINT `no_li`, item c in the list above applies, i.e., some entity data types that are in the supertype constraint are not included in the longform schema;

— applying item c1 results in the following expressions:

```
for p:
     ONEOF(m, box) AND ONEOF(box, a);
for a:
     ONEOF(l, box);
```

— removing the boxes by using the rules in item c2 results in the following supertype constraints:

```
for p:
     ONEOF(m, box) AND ONEOF(box, a);
  => ONEOF(m) AND ONEOF(a);
for a:
     ONEOF(l, box);
  => ONEOF(l);
```

The finally resulting longform is given below:

```
...

ENTITY p
  SUPERTYPE OF (ONEOF(m) AND ONEOF(a));
END_ENTITY;

ENTITY a ABSTRACT SUPERTYPE OF (ONEOF(l))
```



```
  SUBTYPE OF (p);
END_ENTITY;

ENTITY l SUBTYPE OF (a);
END_ENTITY;

ENTITY m SUBTYPE OF (p);
END_ENTITY;

...
```

G.3.4 Conversion of ABSTRACT ENTITY and generalized types used in abstract entities

ABSTRACT ENTITY declarations shall be converted into ABSTRACT SUPERTYPE constraints on the entity in the ISO 10303-11:1994 schema.

An entity may have an attribute that is declared to be of a generalized data type, e.g. an AGGREGATE, a GENERIC_ENTITY, or a SELECT data type that is constrained to select items of type GENERIC_ENTITY. For such attributes in the ABSTRACT ENTITY data type that have a generalized type as their domain the following conversion rules apply:

— if all subtypes redeclare the attribute domain to be of the same type, migrate that type to be the type of the attribute in the SUPERTYPE in the ISO 10303-11:1994 schema;

— in all other cases, create a SELECT data type in the ISO 10303-11:1994 schema and add all named data types that are the types of the attribute in any redeclarations in any SUBTYPE of the SUPERTYPE. The name of the select data type shall be concatenated of the entity data type name of the ABSTRACT SUPERTYPE and the name of the attribute that will be of the select data type. The character `'_'` shall - without the quotes - be placed between the two names. The term `'_select'` shall be added to the name (see `binary_entity_relationship_end_one_select` in the example below). Within the subtypes, this new supertype attribute shall be redeclared to be of the data type of the one select item that it was declared to be in the source schema.

EXAMPLE 1 The `IMPORT` schema uses the entire schema `ABSTRACT_EXAMPLE`.

```
SCHEMA ABSTRACT_EXAMPLE;

ENTITY person;
END_ENTITY;

ENTITY product;
END_ENTITY;

ENTITY organization;
END_ENTITY;

ENTITY binary_entity_relationship ABSTRACT;
  end_one : generic_entity;
  end_two : generic_entity;
END_ENTITY;

ENTITY product_of_organization
  SUBTYPE OF (binary_entity_relationship);
  SELF\binary_entity_relationship.end_one: product;
  SELF\binary_entity_relationship.end_two: organization;
END_ENTITY;

ENTITY person_in_organization
```



```
  SUBTYPE OF (binary_entity_relationship);
  SELF\binary_entity_relationship.end_one: person;
  SELF\binary_entity_relationship.end_two: organization;
END_ENTITY;

END_SCHEMA;


SCHEMA IMPORT;

USE FROM ABSTRACT_EXAMPLE;

END_SCHEMA;
```

During the generation of the longform, the ABSTRACT construct is in the context schema `IMPORT` been converted into a SUPERTYPE constraint. A SELECT data type is introduced to maintain the different data types that the generalized attribute in the supertype resulted in in the subtypes. The following entities are modified compared to the source schema:

```
...

TYPE binary_entity_relationship_end_one_select = SELECT
         (person,
          product);
END_TYPE;

ENTITY binary_entity_relationship
  ABSTRACT SUPERTYPE;
  end_one : binary_entity_relationship_end_one_select;
  end_two : organization;
END_ENTITY;

ENTITY product_of_organization
  SUBTYPE OF (binary_entity_relationship);
  SELF\binary_entity_relationship.end_one: product;
END_ENTITY;

ENTITY person_in_organization
  SUBTYPE OF (binary_entity_relationship);
  SELF\binary_entity_relationship.end_one: person;
END_ENTITY;

...
```

G.3.5 Conversion of attributes renamed in a redeclaration

For each attribute that is RENAMED in a redeclaration in a subtype, the following conversion rules apply:

— the keyword RENAMED and the subsequent new attribute name shall be removed;

— a DERIVE attribute with the new name shall be created in the current SUBTYPE;

— the derivation of the attribute value shall be `SELF\<supertype name>.<old attribute name>`.

EXAMPLE 1 The `IMPORT` schema uses the entire `RENAMED_EXAMPLE` schema.

```
SCHEMA RENAMED_EXAMPLE;

ENTITY binary_entity_relationshp;
   end_one : being;
```



```
   end_two : structure;
END_ENTITY;

ENTITY being;
END_ENTITY;

ENTITY structure;
END_ENTITY;

ENTITY person
  SUBTYPE OF (being);
   role_of_person : STRING;
END_ENTITY;

ENTITY organization
  SUBTYPE OF (structure);
END_ENTITY;

ENTITY person_in_organization
SUBTYPE OF (binary_entity_relationshp);
   SELF\binary_entity_relationshp.end_one RENAMED the_person : person;
   SELF\binary_entity_relationshp.end_two RENAMED the_organization : organization;
END_ENTITY;

END_SCHEMA;


SCHEMA IMPORT;

USE FROM RENAMED_EXAMPLE;

END_SCHEMA;
```

During the generation of the longform, the RENAMED construct is in the context schema `IMPORT` been converted into a DERIVE attribute. Entity `person_in_organization` is modified as follows:

```
...
ENTITY person_in_organization
  SUBTYPE OF (binary_entity_relationship);
  SELF\binary_entity_relationshp.end_two : organization;
   (* THE_ORGANIZATION : This attribute was RENAMED in the ISO~10303-11:2003 source;
      it is remove here, but an attribute with this new name is in the DERIVE clause
      below. *)
  SELF\binary_entity_relationshp.end_one : person;
   (* THE_PERSON : This attribute was RENAMED in the ISO~10303-11:2003 source;
      it is remove here, but an attribute with this new name is in the DERIVE clause
      below. *)
  DERIVE
  the_person : person := SELF\binary_entity_relationship.end_one;
   (* END_ONE : This attribute was RENAMED in the ISO~10303-11:2003 source
              to THE_PERSON.
  the_organization : organization := SELF\binary_entity_relationship.end_two;
   (* END_TWO : This attribute was RENAMED in the ISO~10303-11:2003 source
              to THE_ORGANIZATION.
END_ENTITY;
...
```

NOTE In the case that the attribute name following the RENAMED keyword has anything assigned to it in functions, this will cause failures in those functions. Manual correction of the schema would be

required.

