INTERNATIONAL STANDARD ISO 10303-41:2005
TECHNICAL CORRIGENDUM 1

Published 2008-05-15

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION

Industrial automation systems and integration — Product data representation and exchange —

Part 41: Integrated generic resource: Fundamentals of product description and support

TECHNICAL CORRIGENDUM 1

Systèmes d'automatisation industrielle et intégration — Représentation et échange de données de produits — Partie 41: Ressources génériques intégrées: Principes de description et de support de produits

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to ISO 10303-41:2005 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC 4, *Industrial data*.

Introduction

This Technical Corrigendum applies to ISO 10303-41:2005.

The purpose of the modifications to the text of ISO 10303-41:2005 is to correct a type assignment incompatibility problem by redefining a type definition and moving a related type definition from part 108, to complete the set of date and time definitions, and to correct and complete the set of measure definitions corresponding to SI units.

ICS 25.040.40 **Ref. No. ISO 10303-41:2005/Cor.1:2008(E)**

Published in Switzerland

Modifications to the text of ISO 10303-41:2005

Pages 159 - 171, 16.4 Date and time entity definitions,
The set of date and time entity definitions was incomplete. To add the required definition of month and year make the following amendments to clause 16.4;

Page 161, 16.4.3 date
Replace the first paragraph of clause 16.4.3 with the following:

A **date** is the identification of a day or week or month in a year.

Remove the existing EXPRESS definition and replace with:

EXPRESS specification:

```
 *)
ENTITY date
    SUPERTYPE OF (ONEOF(calendar_date,
                        ordinal_date,
                        week_of_year_and_day_date,
                        year_month));
    year_component : year_number;
END_ENTITY;
(*
```

Page 171, Add the following new clause after clause 16.4.19

16.4.20 year_month

A **year_month** is a type of **date** defined as a month of a year.

EXPRESS specification:

```
*)
ENTITY year_month
  SUBTYPE OF (date);
    month_component : month_in_year_number;
END_ENTITY;
(*
```


Attribute definitions:

month_component: the month element of the date.

Pages 196 - 205, 21.3 Measure type definitions
A **non_negative_length_measure** *type was previously defined in part 108 of ISO 10303 but was not, in the non-zero case, assignment compatible with a* **positive_length_measure** *as defined in this part of ISO 10303. To remove this inconsistency make the following amendments to clause 21.3:*

Page 198, 21.3.11 measure_value
Remove the existing EXPRESS definition and replace with:

EXPRESS specification:

```
*)
  TYPE measure_value = SELECT
    (absorbed_dose_measure,
     acceleration_measure,
     amount_of_substance_measure,
     area_measure,
     capacitance_measure,
     celsius_temperature_measure,
     conductance_measure,
     context_dependent_measure,
     count_measure,
     descriptive_measure,
     dose_equivalent_measure,
     electric_charge_measure,
     electric_current_measure,
     electric_potential_measure,
     energy_measure,
     force_measure,
     frequency_measure,
     illuminance_measure,
     inductance_measure,
     length_measure,
     luminous_flux_measure,
     luminous_intensity_measure,
     magnetic_flux_density_measure,
     magnetic_flux_measure,
     mass_measure,
     numeric_measure,
     non_negative_length_measure,
     parameter_value,
     plane_angle_measure,
     positive_length_measure,
     positive_plane_angle_measure,
     positive_ratio_measure,
```



```
      power_measure,
      pressure_measure,
      radioactivity_measure,
      ratio_measure,
      resistance_measure,
      solid_angle_measure,
      thermodynamic_temperature_measure,
      time_measure,
      velocity_measure,
      volume_measure);
(*
```

Page 199, 21.3.12 numeric_measure
Add after the EXPRESS definition of this type the following note:

NOTE In order to define measure quantities that are not included as specific types in this standard a **numeric_measure** may be used as the **value_component** of a **measure_with_unit**, the corresponding **unit_component** being an **si_unit** or a **derived_unit**.

Page 200, 21.3.15 positive_length_measure
Remove the existing EXPRESS definition and replace with:

EXPRESS specification:

```
*)
  TYPE positive_length_measure = non_negative_length_measure;
    WHERE
      WR1: SELF > 0.0;
  END_TYPE;
(*
```

Page 205, add the following new clauses after clause 21.3.25:

21.3.26 absorbed_dose_measure

An **absorbed_dose_measure** is the value of the absorbed dose of radiation.

EXPRESS specification:

```
*)
  TYPE absorbed_dose_measure = REAL;
  END_TYPE;
```



```
(*
```

21.3.27 acceleration_measure

An **acceleration_measure** is the value of the rate of change of velocity.

EXPRESS specification:

```
*)
  TYPE acceleration_measure = REAL;
  END_TYPE;
(*
```

21.3.28 capacitance_measure

A **capacitance_measure** is the value of capacitance.

EXPRESS specification:

```
*)
  TYPE capacitance_measure = REAL;
  END_TYPE;
(*
```

21.3.29 conductance_measure

A **conductance_measure** is the value of an electrical conductance.

EXPRESS specification:

```
*)
  TYPE conductance_measure = REAL;
  END_TYPE;
(*
```

21.3.30 dose_equivalent_measure

A **dose_equivalent_measure** is the value of the radiation dose equivalent.

EXPRESS specification:

```
*)
  TYPE radioactivity_measure = REAL;
  END_TYPE;
(*
```

21.3.31 electric_charge_measure

An **electric_charge_measure** is the value of an electrical charge.

EXPRESS specification:

```
*)
  TYPE electric_charge_measure = REAL;
  END_TYPE;
(*
```

21.3.32 electric_potential_measure

An **electric_potential_measure** is the value of an electrical potential.

EXPRESS specification:

```
*)
  TYPE electric_potential_measure = REAL;
  END_TYPE;
(*
```

21.3.33 energy_measure

An **energy_measure** is the value of energy, or work done, in a system.

EXPRESS specification:

```
*)
  TYPE energy_measure = REAL;
  END_TYPE;
(*
```


21.3.34 force_measure

A **force_measure** is the value of a force.

EXPRESS specification:

```
*)
  TYPE force_measure = REAL;
  END_TYPE;
(*
```

21.3.35 frequency_measure

A **frequency_measure** is the value of a frequency.

EXPRESS specification:

```
*)
  TYPE frequency_measure = REAL;
  END_TYPE;
(*
```

21.3.36 illuminance_measure

An **illuminance_measure** is the value of illuminance.

EXPRESS specification:

```
*)
  TYPE illuminance_measure = REAL;
  END_TYPE;
(*
```

21.3.37 inductance_measure

An **inductance_measure** is the value of inductance.

EXPRESS specification:

```
*)
  TYPE inductance_measure = REAL;
  END_TYPE;
(*
```

21.3.38 luminous_flux_measure

A **luminous_flux_measure** is the value of luminous flux.

EXPRESS specification:

```
*)
  TYPE luminous_flux_measure = REAL;
  END_TYPE;
(*
```

21.3.39 magnetic_flux_density_measure

A **magnetic_flux_density_measure** is the value of magnetic flux density.

EXPRESS specification:

```
*)
  TYPE magnetic_flux_density_measure = REAL;
  END_TYPE;
(*
```

21.3.40 magnetic_flux_measure

A **magnetic_flux_measure** is the value of magnetic flux.

EXPRESS specification:

```
*)
  TYPE magnetic_flux_measure = REAL;
  END_TYPE;
(*
```


21.3.41 non_negative_length_measure

A **non_negative_length_measure** type is a **length_measure** whose value is greater than or equal to zero.

EXPRESS specification:

```
*)
TYPE non_negative_length_measure = length_measure;
 WHERE
  WR1: SELF >= 0.0;
END_TYPE;
(*
```

Formal propositions:

WR1: A **non_negative_length_measure** shall be positive or zero.

21.3.42 power_measure

A **power_measure** is the value of power, or the rate of doing work.

EXPRESS specification:

```
*)
  TYPE power_measure = REAL;
  END_TYPE;
(*
```

21.3.43 pressure_measure

A **pressure_measure** is the value of force per unit area.

EXPRESS specification:

```
*)
  TYPE pressure_measure = REAL;
  END_TYPE;
(*
```


21.3.44 radioactivity_measure

A **radioactivity_measure** is the value of the radioactive disintegration.

EXPRESS specification:

```
*)
  TYPE radioactivity_measure = REAL;
  END_TYPE;
(*
```

21.3.45 resistance_measure

A **resistance_measure** is the value of electrical resistance.

EXPRESS specification:

```
*)
  TYPE resistance_measure = REAL;
  END_TYPE;
(*
```

21.3.46 velocity_measure

A **velocity_measure** is the value of the rate of change of position.

EXPRESS specification:

```
*)
  TYPE velocity_measure = REAL;
  END_TYPE;
(*
```

Page 207, 21.4.4 area_unit

The original definition was deprecated since it was incorrectly defined and could not be used. Remove the current clause 21.4.4 and replace with:

21.4.4 area_unit

An **area_unit** is a type of **derived_unit** used to measure the extent of a surface.

EXPRESS specification:

```
*)
  ENTITY area_unit
   SUBTYPE OF (derived_unit);
  WHERE
  WR1: derive_dimensional_exponents(SELF) =
      dimensional_exponents ( 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
  END_ENTITY;
 (*
```

Formal propositions:

WR1: The dimensional exponent of length shall be equal to two and all the other dimensional exponents shall be equal to zero.

Page 208, 21.4.7 conversion_based__unit

The original definition of this entity required the user to define the value of the inherited attribute dimensions. The result could be inconsistent with the dimensions of the conversion_factor. Remove the current EXPRESS definition and replace with:

EXPRESS specification:

```
*)
ENTITY conversion_based_unit
  SUBTYPE OF (named_unit);
  name : label;
  conversion_factor : measure_with_unit;
  DERIVE
   SELF\named_unit.dimensions : dimensional_exponents :=
   derive_dimensional_exponents(conversion_factor\measure_with_unit.unit_component);
END_ENTITY;
 (*
```

Page 209, 21.4.8 derived_unit

New subtypes of this entity have been defined, remove the current EXPRESS definition and replace with:

EXPRESS specification:

```
*)
  ENTITY derived_unit;
   SUPERTYPE OF (ONEOF(absorbed_dose_unit,
    acceleration_unit,
    capacitance_unit,
    area_unit,
    capacitance_unit,
    conductance_unit,
    dose_equivalent_unit,
    electric_charge_unit,
    electric_potential_unit,
    energy_unit,
    force_unit,
    frequency_unit,
    illuminance_unit,
    inductance_unit,
    magnetic_flux_density_unit,
    magnetic_flux_unit,
    power_unit,
    pressure_unit,
    radioactivity_unit,
    resistance_unit,
    velocity_unit,
    volume_unit));
    elements : SET [1:?] OF derived_unit_element;
  DERIVE
    name : label := get_name_value(SELF);
  WHERE
    WR1: (SIZEOF(elements) > 1) OR ((SIZEOF(elements) = 1) AND
    (elements[1].exponent <> 1.0));
    WR2: SIZEOF(USEDIN(SELF, 'BASIC_ATTRIBUTE_SCHEMA.' +
           'NAME_ATTRIBUTE.NAMED_ITEM')) <= 1;
  END_ENTITY;
 (*
```

Page 215, 21.4.20 measure_with_unit

The list of subtypes of this entity has been extended, remove the current EXPRESS definition and replace with:

EXPRESS specification:

```
*)
  ENTITY measure_with_unit
    SUPERTYPE OF (ONEOF(length_measure_with_unit, mass_measure_with_unit,
    time_measure_with_unit,     electric_current_measure_with_unit,
```

```
    thermodynamic_temperature_measure_with_unit,
    celsius_temperature_measure_with_unit,
    amount_of_substance_measure_with_unit, luminous_intensity_measure_with_unit,
    plane_angle_measure_with_unit, solid_angle_measure_with_unit,
    area_measure_with_unit, volume_measure_with_unit, ratio_measure_with_unit,
    absorbed_dose_measure_with_unit,
    acceleration_measure_with_unit,
    capacitance_measure_with_unit,
    conductance_measure_with_unit,
    dose_equivalent_measure_with_unit,
    electric_charge_measure_with_unit,
    electric_potential_measure_with_unit,
    energy_measure_with_unit,
    force_measure_with_unit,
    frequency_measure_with_unit,
    illuminance_measure_with_unit,
    inductance_measure_with_unit,
    luminous_flux_measure_with_unit,
    magnetic_flux_density_measure_with_unit,
    magnetic_flux_measure_with_unit,
    power_measure_with_unit,
    pressure_measure_with_unit,
    radioactivity_measure_with_unit,
    resistance_measure_with_unit,
    velocity_measure_with_unit));
    value_component : measure_value;
    unit_component : unit;
  WHERE
    WR1: valid_units(SELF);
  END_ENTITY;
(*
```

Page 216, 21.4.21 named_unit

The list of subtypes of this entity has been changed, remove the current EXPRESS definition and replace with:

EXPRESS specification:

```
*)
  ENTITY named_unit
    SUPERTYPE OF (ONEOF(si_unit, conversion_based_unit, context_dependent_unit)
    ANDOR ONEOF(length_unit, mass_unit, time_unit, electric_current_unit,
    thermodynamic_temperature_unit, amount_of_substance_unit,
    luminous_flux_unit, luminous_intensity_unit,
    plane_angle_unit, solid_angle_unit, ratio_unit));
    dimensions : dimensional_exponents;
  END_ENTITY;
(*
```


Page 222, 21.4.34 volume_unit

The original definition was deprecated since it was incorrectly defined and could not be used. Remove the current clause 21.4.34 and replace with:

21.4.34 volume_unit

A **volume_unit** is a type of **derived_unit** used to measure the extent of a solid.

EXPRESS specification:

```
*)
  ENTITY volume_unit
    SUBTYPE OF (derived_unit);
   WHERE
     WR1: derive_dimensional_exponents(SELF) =
          dimensional_exponents ( 3.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 );
  END_ENTITY;
(*
```

Formal propositions:

WR1: The dimensional exponent of length shall be equal to three and all the other dimensional exponents shall be equal to zero.

Page 223, before 21.5 Measure function definitions

The set of unit definitions has been extended to include units corresponding to most of the standard SI units. Add the following new clauses between the end of 21.4.34 and the start of 21.5:

21.4.35 absorbed_dose_measure_with_unit

An **absorbed_dose_measure_with_unit** is a type of **measure_with_unit** in which the physical quantity is an absorbed dose of radiation as defined in ISO 31.

EXPRESS specification:

```
*)
  ENTITY absorbed_dose_measure_with_unit
```



```
    SUBTYPE OF (measure_with_unit);
  WHERE
    WR1: 'MEASURE_SCHEMA.ABSORBED_DOSE_UNIT' IN
              TYPEOF(SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **unit_component** be of type **absorbed_dose_unit**.

21.4.36 absorbed_dose_unit

An **absorbed_dose_unit** is a type of **derived_unit** in which the absorbed dose is expressed.

EXPRESS specification:

```
*)
  ENTITY absorbed_dose_unit
    SUBTYPE OF(derived_unit);
  WHERE
    WR1: derive_dimensional_exponents(SELF) =
              dimensions_for_si_unit (si_unit_name.gray);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The dimensional exponents shall be equal to those of the Gray as defined in ISO 31.

21.4.37 si_absorbed_dose_unit

An **si_absorbed_dose_unit** is a type of **absorbed_dose_unit** in which the absorbed dose is expressed in SI units.

EXPRESS specification:

```
*)
  ENTITY si_absorbed_dose_unit
     SUBTYPE OF (absorbed_dose_unit,si_unit);
  WHERE
```



```
    WR1: SELF\si_unit.name = si_unit_name.gray;
    WR2: NOT EXISTS(SELF\derived_unit.name);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **name** of the **si_unit** shall be farad.

WR2: No other name shall be assigned to this entity.

21.4.38 acceleration_measure_with_unit

An **acceleration_measure_with_unit** is a type of **measure_with_unit** in which the physical quantity is an acceleration.

EXPRESS specification:

```
*)
  ENTITY acceleration_measure_with_unit
    SUBTYPE OF (measure_with_unit);
  WHERE
    WR1: 'MEASURE_SCHEMA.ACCELERATION_UNIT' IN
          TYPEOF (SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **unit_component** be of type **acceleration_unit**.

21.4.39 acceleration_unit

An **acceleration_unit** is a type of **derived_unit** in which the acceleration of an object is expressed.

EXPRESS specification:

```
*)
  ENTITY acceleration_unit
    SUBTYPE OF (derived_unit);
  WHERE
```



```
    WR1: derive_dimensional_exponents(SELF) =
       dimensional_exponents ( 1.0, 0.0, -2.0, 0.0, 0.0, 0.0, 0.0 );
  END_ENTITY;
(*
```

Formal propositions:

WR1: The dimensional exponent of length shall be equal to one, the dimensional exponent of time shall be equal to minus two, and all the other dimensional exponents shall be equal to zero.

21.4.40 capacitance_measure_with_unit

A **capacitance_measure_with_unit** is a type of **measure_with_unit** in which the physical quantity is an electric capacitance as defined in ISO 31.

EXPRESS specification:

```
*)
  ENTITY capacitance_measure_with_unit
    SUBTYPE OF (measure_with_unit);
  WHERE
    WR1: 'MEASURE_SCHEMA.CAPACITANCE_UNIT' IN
              TYPEOF(SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **unit_component** be of type **capacitance_unit**.

21.4.41 capacitance_unit

A **capacitance_unit** is a type of **derived_unit** in which the capacitance is expressed.

EXPRESS specification:

```
*)
  ENTITY capacitance_unit
    SUBTYPE OF(derived_unit);
  WHERE
    WR1: derive_dimensional_exponents(SELF) =
```



```
            dimensions_for_si_unit (si_unit_name.farad);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The dimensional exponents shall be equal to those of the farad as defined in ISO 31.

21.4.42 si_capacitance_unit

An **si_capacitance_unit** is a type of **capacitance_unit** in which the electric capacitance is expressed in SI units.

EXPRESS specification:

```
*)
  ENTITY si_capacitance_unit
     SUBTYPE OF (capacitance_unit,si_unit);
  WHERE
    WR1: SELF\si_unit.name = si_unit_name.farad;
    WR2: NOT EXISTS(SELF\derived_unit.name);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **name** of the **si_unit** shall be farad.

WR2: No other name shall be assigned to this entity.

21.4.43 conductance_measure_with_unit

A **conductance_measure_with_unit** is a type of **measure_with_unit** in which the physical quantity is a conductance as defined in ISO 31.

EXPRESS specification:

```
*)
  ENTITY conductance_measure_with_unit
    SUBTYPE OF (measure_with_unit);
  WHERE
```



```
    WR1: 'MEASURE_SCHEMA.CONDUCTANCE_UNIT' IN
              TYPEOF(SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **unit_component** shall be of type **conductance_unit**.

21.4.44 conductance_unit

A **conductance_unit** is a type of **derived_unit** in which the electrical conductance is expressed.

EXPRESS specification:

```
*)
   ENTITY conductance_unit
    SUBTYPE OF(derived_unit);
  WHERE
    WR1: derive_dimensional_exponents(SELF) =
           dimensions_for_si_unit (si_unit_name.siemens);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The dimensional exponents shall be equal to those of the siemens as defined in ISO 31.

21.4.45 si_conductance_unit

An **si_conductance_unit** is a type of **conductance_unit** in which the electrical conductance is expressed in SI units.

EXPRESS specification:

```
*)
  ENTITY si_conductance_unit
     SUBTYPE OF (conductance_unit,si_unit);
  WHERE
    WR1: SELF\si_unit.name = si_unit_name.siemens;
    WR2: NOT EXISTS(SELF\derived_unit.name);
```



```
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **name** of the **si_unit** shall be siemens.

WR2: No other name shall be assigned to this entity.

21.4.46 dose_equivalent_measure_with_unit

A **dose_equivalent_measure_with_unit** is a type of **measure_with_unit** in which the physical quantity is a radiation dose equivalent as defined in ISO 31.

EXPRESS specification:

```
*)
  ENTITY dose_equivalent_measure_with_unit
    SUBTYPE OF (measure_with_unit);
  WHERE
    WR1: 'MEASURE_SCHEMA.DOSE_EQUIVALENT_UNIT' IN
                TYPEOF(SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **unit_component** shall be of type **dose_equivalent_unit**.

21.4.47 dose_equivalent_unit

A **dose_equivalent_unit** is a type of **derived_unit** in which the radiation dose equivalent is expressed.

EXPRESS specification:

```
*)
   ENTITY dose\_equivalent_unit
    SUBTYPE OF(derived_unit);
  WHERE
    WR1: derive_dimensional_exponents(SELF) =
              dimensions_for_si_unit (si_unit_name.sievert);
```



```
  END_ENTITY;
(*
```

Formal propositions:

WR1: The dimensional exponents shall be equal to those of the sievert as defined in ISO 31.

21.4.48 si_dose_equivalent_unit

An **si_dose_equivalent_unit** is a type of **dose_equivalent_unit** in which the radiation dose equivalent is expressed in SI units.

EXPRESS specification:

```
*)
  ENTITY si_dose\_equivalent_unit
     SUBTYPE OF (dose\_equivalent_unit,si_unit);
  WHERE
    WR1: SELF\si_unit.name = si_unit_name.sievert;
    WR2: NOT EXISTS(SELF\derived_unit.name);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **name** of the **si_unit** shall be sievert.

WR2: No other name shall be assigned to this entity.

21.4.49 electric_charge_measure_with_unit

An **electric_charge_measure_with_unit** is a type of **measure_with_unit** in which the physical quantity is an electric charge as defined in ISO 31.

EXPRESS specification:

```
*)
  ENTITY electric_charge_measure_with_unit
    SUBTYPE OF (measure_with_unit);
  WHERE
    WR1: 'MEASURE_SCHEMA.ELECTRIC_CHARGE_UNIT' IN
```



```
                TYPEOF(SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **unit_component** shall be of type **electric_charge_unit**.

21.4.50 electric_charge_unit

An **electric_charge_unit** is a type of **derived_unit** in which the electric charge is expressed.

EXPRESS specification:

```
*)
  ENTITY electric_charge_unit
    SUBTYPE OF(derived_unit);
  WHERE
    WR1: derive_dimensional_exponents(SELF) =
       dimensions_for_si_unit (si_unit_name.coulomb);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The dimensional exponents shall be equal to those of the coulomb as defined in ISO 31.

21.4.51 si_electric_charge_unit

An **si_electric_charge_unit** is a type of **electric_charge_unit** in which the electric charge is expressed in SI units.

EXPRESS specification:

```
*)
  ENTITY si_electric_charge_unit
     SUBTYPE OF (electric_charge_unit,si_unit);
  WHERE
    WR1: SELF\si_unit.name = si_unit_name.volt;
    WR2: NOT EXISTS(SELF\derived_unit.name);
  END_ENTITY;
(*
```


Formal propositions:

WR1: The **name** of the **si_unit** shall be volt.

WR2: No other name shall be assigned to this entity.

21.4.52 electric_potential_measure_with_unit

An **electric_potential_measure_with_unit** is a type of **measure_with_unit** in which the physical quantity is an electric potential as defined in ISO 31.

EXPRESS specification:

```
*)
  ENTITY electric_potential_measure_with_unit
    SUBTYPE OF (measure_with_unit);
  WHERE
    WR1: 'MEASURE_SCHEMA.ELECTRIC_POTENTIAL_UNIT' IN
              TYPEOF(SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **unit_component** shall be of type **electric_potential_unit**.

21.4.53 electric_potential_unit

An **electric_potential_unit** is a type of **derived_unit** in which the electric potential is expressed.

EXPRESS specification:

```
*)
  ENTITY electric_potential_unit
    SUBTYPE OF(derived_unit);
  WHERE
    WR1: derive_dimensional_exponents(SELF) =
       dimensions_for_si_unit (si_unit_name.volt);
  END_ENTITY;

(*
```


Formal propositions:

WR1: The dimensional exponents shall be equal to those of the volt as defined in ISO 31.

21.4.54 si_electric_potential_unit

An **si_electric_potential_unit** is a type of **electric_potential_unit** in which the electric potential is expressed in SI units.

EXPRESS specification:

```
*)
  ENTITY si_electric_potential_unit
     SUBTYPE OF (electric_potential_unit,si_unit);
  WHERE
    WR1: SELF\si_unit.name = si_unit_name.volt;
    WR2: NOT EXISTS(SELF\derived_unit.name);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **name** of the **si_unit** shall be volt.

WR2: No other name shall be assigned to this entity.

21.4.55 energy_measure_with_unit

An **energy_measure_with_unit** is a type of **measure_with_unit** in which the physical quantity is energy as defined in ISO 31.

EXPRESS specification:

```
*)
  ENTITY energy_measure_with_unit
    SUBTYPE OF (measure_with_unit);
  WHERE
    WR1: 'MEASURE_SCHEMA.ENERGY_UNIT' IN
           TYPEOF(SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```


Formal propositions:

WR1: The **unit_component** shall be of type **energy_unit**.

21.4.56 energy_unit

An **energy_unit** is a type of **derived_unit** in which the energy is expressed.

EXPRESS specification:

```
*)
  ENTITY energy_unit
    SUBTYPE OF(derived_unit);
  WHERE
    WR1: derive_dimensional_exponents(SELF) =
              dimensions_for_si_unit (si_unit_name.joule);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The dimensional exponents shall be equal to those of the joule as defined in ISO 31.

21.4.57 si_energy_unit

An **si_energy_unit** is a type of **energy_unit** in which the energy is expressed in SI units.

EXPRESS specification:

```
*)
  ENTITY si_energy_unit
     SUBTYPE OF (energy_unit,si_unit);
  WHERE
    WR1: SELF\si_unit.name = si_unit_name.joule;
    WR2: NOT EXISTS(SELF\derived_unit.name);
  END_ENTITY;
(*
```


Formal propositions:

WR1: The **name** of the **si_unit** shall be joule.

WR2: No other name shall be assigned to this entity.

21.4.58 force_measure_with_unit

A **force_measure_with_unit** is a type of **measure_with_unit** in which the physical quantity is force as defined in ISO 31.

EXPRESS specification:

```
*)
  ENTITY force_measure_with_unit
    SUBTYPE OF (measure_with_unit);
  WHERE
    WR1: 'MEASURE_SCHEMA.FORCE_UNIT' IN
              TYPEOF(SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **unit_component** shall be of type **force_unit**.

21.4.59 force_unit

A **force_unit** is a type of **derived_unit** in which the force is expressed.

EXPRESS specification:

```
*)
  ENTITY force_unit
    SUBTYPE OF(derived_unit);
  WHERE
    WR1: derive_dimensional_exponents(SELF) =
              dimensions_for_si_unit (si_unit_name.newton);
  END_ENTITY;
(*
```


Formal propositions:

WR1: The dimensional exponents shall be equal to those of the newton as defined in ISO 31.

21.4.60 si_force_unit

An **si_force_unit** is a type of **force_unit** in which the force is expressed in SI units.

EXPRESS specification:

```
*)
  ENTITY si_force_unit
     SUBTYPE OF (force_unit,si_unit);
  WHERE
    WR1: SELF\si_unit.name = si_unit_name.newton;
    WR2: NOT EXISTS(SELF\derived_unit.name);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **name** of the **si_unit** shall be newton.

WR2: No other name shall be assigned to this entity.

21.4.61 frequency_measure_with_unit

A **frequency_measure_with_unit** is a type of **measure_with_unit** in which the quantity measured is a frequency as defined in ISO 31.

EXPRESS specification:

```
*)
  ENTITY frequency_measure_with_unit
    SUBTYPE OF (measure_with_unit);
  WHERE
    WR1: 'MEASURE_SCHEMA.FREQUENCY_UNIT' IN
                TYPEOF(SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```


Formal propositions:

WR1: The **unit_component** shall be of type **frequency_unit**.

21.4.62 frequency_unit

A **frequency_unit** is a type of **derived_unit** in which the frequency is expressed.

EXPRESS specification:

```
*)
  ENTITY frequency_unit
    SUBTYPE OF(derived_unit);
  WHERE
    WR1: derive_dimensional_exponents(SELF) =
              dimensions_for_si_unit (si_unit_name.hertz);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The dimensional exponents shall be equal to those of the hertz as defined in ISO 31.

21.4.63 si_frequency_unit

An **si_frequency_unit** is a type of **frequency_unit** in which the frequency is expressed in SI units.

EXPRESS specification:

```
*)
  ENTITY si_frequency_unit
     SUBTYPE OF (frequency_unit,si_unit);
  WHERE
    WR1: SELF\si_unit.name = si_unit_name.hertz;
    WR2: NOT EXISTS(SELF\derived_unit.name);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **name** of the **si_unit** shall be hertz.

WR2: No other name shall be assigned to this entity.

21.4.64 illuminance_measure_with_unit

An **illuminance_measure_with_unit** is a type of **measure_with_unit** in which the quantity measured is an illuminance as defined in ISO 31.

EXPRESS specification:

```
*)
  ENTITY illuminance_measure_with_unit
    SUBTYPE OF (measure_with_unit);
  WHERE
    WR1: 'MEASURE_SCHEMA.ILLUMINANCE_UNIT' IN
              TYPEOF(SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **unit_component** shall be of type **illuminance_unit**.

21.4.65 illuminance_unit

An **illuminance_unit** is a type of **derived_unit** in which the illuminance is expressed.

EXPRESS specification:

```
*)
  ENTITY illuminance_unit
    SUBTYPE OF(derived_unit);
  WHERE
    WR1: derive_dimensional_exponents(SELF) =
              dimensions_for_si_unit (si_unit_name.lux);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The dimensional exponents shall be equal to those of the lux as defined in ISO 31.

21.4.66 si_illuminance_unit

An **si_illuminance_unit** is a type of **illuminance_unit** in which the illuminance is expressed in SI units.

EXPRESS specification:

```
*)
  ENTITY si_illuminance_unit
     SUBTYPE OF (illuminance_unit,si_unit);
  WHERE
    WR1: SELF\si_unit.name = si_unit_name.lux;
    WR2: NOT EXISTS(SELF\derived_unit.name);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **name** of the **si_unit** shall be lux.

WR2: No other name shall be assigned to this entity.

21.4.67 inductance_measure_with_unit

An **inductance_measure_with_unit** is a type of **measure_with_unit** in which the quantity measured is an inductance as defined in ISO 31.

EXPRESS specification:

```
*)
  ENTITY inductance_measure_with_unit
    SUBTYPE OF (measure_with_unit);
  WHERE
    WR1: 'MEASURE_SCHEMA.INDUCTANCE_UNIT' IN
              TYPEOF(SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **unit_component** shall be of type **inductance_unit**.

21.4.68 inductance_unit

An **inductance_unit** is a type of **derived_unit** in which the inductance is expressed.

EXPRESS specification:

```
*)
  ENTITY inductance_unit
    SUBTYPE OF(derived_unit);
  WHERE
    WR1: derive_dimensional_exponents(SELF) =
           dimensions_for_si_unit (si_unit_name.henry);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The dimensional exponents shall be equal to those of the henry as defined in ISO 31.

21.4.69 si_inductance_unit

An **si_inductance_unit** is a type of **inductance_unit** in which the inductance is expressed in SI units.

EXPRESS specification:

```
*)
  ENTITY si_inductance_unit
     SUBTYPE OF (inductance_unit,si_unit);
  WHERE
    WR1: SELF\si_unit.name = si_unit_name.henry;
    WR2: NOT EXISTS(SELF\derived_unit.name);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **name** of the **si_unit** shall be henry.

WR2: No other name shall be assigned to this entity.

21.4.70 luminous_flux_measure_with_unit

A **luminous_flux_measure_with_unit** is a type of **named_unit** in which the quantity measured is a luminous flux as defined in ISO 31.

EXPRESS specification:

```
*)
  ENTITY luminous_flux_measure_with_unit
    SUBTYPE OF (measure_with_unit);
  WHERE
    WR1: 'MEASURE_SCHEMA.LUMINOUS_FLUX_UNIT' IN
              TYPEOF(SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **unit_component** shall be of type **luminous_flux_unit**.

21.4.71 luminous_flux_unit

A **luminous_flux_unit** is a type of **derived_unit** in which the luminous flux is expressed.

EXPRESS specification:

```
*)
  ENTITY luminous_flux_unit
    SUBTYPE OF (named_unit);
  WHERE
    WR1: derive_dimensional_exponents(SELF) =
                  dimensions_for_si_unit (si_unit_name.lumen);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The dimensional exponents shall be equal to those of the lumen as defined in ISO 31.

21.4.72 magnetic_flux_density_measure_with_unit

A **magnetic_flux_density_measure_with_unit** is a type of **measure_with_unit** in which the physical quantity is magnetic flux density as defined in ISO 31.

EXPRESS specification:

```
*)
  ENTITY magnetic_flux_density_measure_with_unit
    SUBTYPE OF (measure_with_unit);
  WHERE
    WR1: 'MEASURE_SCHEMA.MANETIC_FLUX_DENSITY_UNIT' IN
              TYPEOF(SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **unit_component** shall be of type **magnetic_flux_density_unit**.

21.4.73 magnetic_flux_density_unit

A **magnetic_flux_density_unit** is a type of **derived_unit** in which the magnetic flux density is expressed.

EXPRESS specification:

```
*)
  ENTITY magnetic_flux_density_unit
    SUBTYPE OF(derived_unit);
  WHERE
    WR1: derive_dimensional_exponents(SELF) =
                  dimensions_for_si_unit (si_unit_name.tesla);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The dimensional exponents shall be equal to those of the tesla as defined in ISO 31.

21.4.74 si_magnetic_flux_density_unit

An **si_magnetic_flux_density_unit** is a type of **magnetic_flux_density_unit** in which the magnetic flux density is expressed in SI units.

EXPRESS specification:

```
*)
  ENTITY si_magnetic_flux_density_unit
     SUBTYPE OF (magnetic_flux_density_unit,si_unit);
  WHERE
    WR1: SELF\si_unit.name = si_unit_name.tesla;
    WR2: NOT EXISTS(SELF\derived_unit.name);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **name** of the **si_unit** shall be tesla.

WR2: No other name shall be assigned to this entity.

21.4.75 magnetic_flux_measure_with_unit

A **magnetic_flux_measure_with_unit** is a type of **measure_with_unit** in which the quantity measured is a magnetic flux as defined in ISO 31.

EXPRESS specification:

```
*)
  ENTITY magnetic_flux_measure_with_unit
    SUBTYPE OF (measure_with_unit);
  WHERE
    WR1: 'MEASURE_SCHEMA.MAGNETIC_FLUX_UNIT' IN
              TYPEOF(SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **unit_component** shall be of type **magnetic_flux_unit**.

21.4.76 magnetic_flux_unit

A **magnetic_flux_unit** is a type of **derived_unit** in which the magnetic flux is expressed.

EXPRESS specification:

```
*)
  ENTITY magnetic_flux_unit
    SUBTYPE OF(derived_unit);
  WHERE
    WR1: derive_dimensional_exponents(SELF) =
              dimensions_for_si_unit (si_unit_name.weber);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The dimensional exponents shall be equal to those of the weber as defined in ISO 31.

21.4.77 si_magnetic_flux_unit

An **si_magnetic_flux_unit** is a type of **magnetic_flux_unit** in which the magnetic flux is expressed in SI units.

EXPRESS specification:

```
*)
  ENTITY si_magnetic_flux_unit
     SUBTYPE OF (magnetic_flux_unit,si_unit);
  WHERE
    WR1: SELF\si_unit.name = si_unit_name.weber;
    WR2: NOT EXISTS(SELF\derived_unit.name);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **name** of the **si_unit** shall be weber.

WR2: No other name shall be assigned to this entity.

21.4.78 power_measure_with_unit

A **power_measure_with_unit** is a type of **measure_with_unit** in which the quantity measured is a power as defined in ISO 31.

EXPRESS specification:

```
*)
  ENTITY power_measure_with_unit
    SUBTYPE OF (measure_with_unit);
  WHERE
    WR1: 'MEASURE_SCHEMA.POWER_UNIT' IN
                TYPEOF(SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **unit_component** shall be of type **power_unit**.

21.4.79 power_unit

A **power_unit** is a type of **derived_unit** in which the power is expressed.

EXPRESS specification:

```
*)
  ENTITY power_unit
    SUBTYPE OF(derived_unit);
  WHERE
    WR1: derive_dimensional_exponents(SELF) =
              dimensions_for_si_unit (si_unit_name.watt);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The dimensional exponents shall be equal to those of the watt as defined in ISO 31.

21.4.80 si_power_unit

An **si_power_unit** is a type of **power_unit** in which the power is expressed in SI units.

EXPRESS specification:

```
*)
  ENTITY si_power_unit
     SUBTYPE OF (power_unit,si_unit);
  WHERE
    WR1: SELF\si_unit.name = si_unit_name.watt;
    WR2: NOT EXISTS(SELF\derived_unit.name);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **name** of the **si_unit** shall be watt.

WR2: No other name shall be assigned to this entity.

21.4.81 pressure_measure_with_unit

A **pressure_measure_with_unit** is a type of **measure_with_unit** in which the quantity measured is a pressure as defined in ISO 31.

EXPRESS specification:

```
*)
  ENTITY pressure_measure_with_unit
    SUBTYPE OF (measure_with_unit);
  WHERE
    WR1: 'MEASURE_SCHEMA.PRESSURE_UNIT' IN
              TYPEOF(SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **unit_component** shall be of type **pressure_unit**.

21.4.82 pressure_unit

A **pressure_unit** is a type of **derived_unit** in which the pressure is expressed.

EXPRESS specification:

```
*)
  ENTITY pressure_unit
    SUBTYPE OF(derived_unit);
  WHERE
    WR1: derive_dimensional_exponents(SELF) =
              dimensions_for_si_unit (si_unit_name.pascal);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The dimensional exponents shall be equal to those of the pascal as defined in ISO 31.

21.4.83 si_pressure_unit

An **si_pressure_unit** is a type of **pressure_unit** in which the pressure is expressed in SI units.

EXPRESS specification:

```
*)
  ENTITY si_pressure_unit
     SUBTYPE OF (pressure_unit\si_unit);
  WHERE
    WR1: SELF\si_unit.name = si_unit_name.pascal;
    WR2: NOT EXISTS(SELF\derived_unit.name);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **name** of the **si_unit** shall be pascal.

WR2: No other name shall be assigned to this entity.

21.4.84 radioactivity_measure_with_unit

A **radioactivity_measure_with_unit** is a type of **measure_with_unit** in which the physical quantity is radioactivity as defined in ISO 31.

EXPRESS specification:

```
*)
  ENTITY radioactivity_measure_with_unit
    SUBTYPE OF (measure_with_unit);
  WHERE
    WR1: 'MEASURE_SCHEMA.radioactivity_UNIT' IN
                TYPEOF(SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **unit_component** shall be of type **radioactivity_unit**.

21.4.85 radioactivity_unit

A **radioactivity_unit** is a type of **derived_unit** in which the radioactivity is expressed.

EXPRESS specification:

```
*)
   ENTITY radioactivity_unit
    SUBTYPE OF(derived_unit);
  WHERE
    WR1: derive_dimensional_exponents(SELF) =
              dimensions_for_si_unit (si_unit_name.becquerel);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The dimensional exponents shall be equal to those of the becquerel as defined in ISO 31.

21.4.86 si_radioactivity_unit

An **si_radioactivity_unit** is a type of **electric_radioactivity_unit** in which the radioactivity is expressed in SI units.

EXPRESS specification:

```
*)
  ENTITY si_radioactivity_unit
     SUBTYPE OF (radioactivity_unit,si_unit);
  WHERE
    WR1: SELF\si_unit.name = si_unit_name.becquerel;
    WR2: NOT EXISTS(SELF\derived_unit.name);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **name** of the **si_unit** shall be becquerel.

WR2: No other name shall be assigned to this entity.

21.4.87 resistance_measure_with_unit

A **resistance_measure_with_unit** is a type of **measure_with_unit** in which the quantity measured is an electrical resistance as defined in ISO 31.

EXPRESS specification:

```
*)
  ENTITY resistance_measure_with_unit
    SUBTYPE OF (measure_with_unit);
  WHERE
    WR1: 'MEASURE_SCHEMA.RESISTANCE_UNIT' IN
            TYPEOF(SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **unit_component** shall be of type **resistance_unit**.

21.4.88 resistance_unit

A **resistance_unit** is a type of **derived_unit** in which the electrical resistance is expressed.

EXPRESS specification:

```
*)
  ENTITY resistance_unit
    SUBTYPE OF(derived_unit);
  WHERE
    WR1: derive_dimensional_exponents(SELF) =
            dimensions_for_si_unit (si_unit_name.ohm);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The dimensional exponents shall be equal to those of the ohm as defined in ISO 31.

21.4.89 si_resistance_unit

An **si_resistance_unit** is a type of **resistance_unit** in which the resistance is expressed in SI units.

EXPRESS specification:

```
*)
  ENTITY si_resistance_unit
     SUBTYPE OF (resistance_unit,si_unit);
  WHERE
    WR1: SELF\si_unit.name = si_unit_name.ohm;
    WR2: NOT EXISTS(SELF\derived_unit.name);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **name** of the **si_unit** shall be ohm.

WR2: No other name shall be assigned to this entity.

21.4.90 velocity_measure_with_unit

A **velocity_measure_with_unit** is a type of **measure_with_unit** in which the quantity measured is a velocity.

EXPRESS specification:

```
*)
  ENTITY velocity_measure_with_unit
   SUBTYPE OF (measure_with_unit);
  WHERE
   WR1: 'MEASURE_SCHEMA.VELOCITY_UNIT' IN
            TYPEOF (SELF\measure_with_unit.unit_component);
  END_ENTITY;
(*
```

Formal propositions:

WR1: The **unit_component** shall be of type **velocity_unit**.

21.4.91 velocity_unit

A **velocity_unit** is a type of **derived_unit** in which the velocity is expressed.

EXPRESS specification:

```
*)
  ENTITY velocity_unit
    SUBTYPE OF (derived_unit);
  WHERE
    WR1: derive_dimensional_exponents(SELF) =
          dimensional_exponents ( 1.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0 );
  END_ENTITY;
(*
```

Formal propositions:

WR1: The dimensional exponent of length shall be equal to one, the dimensional exponent of time shall be equal to minus one, and all the other dimensional exponents shall be equal to zero.

Page 223, 21.5.1 derive_dimensional_exponents

The function **derive_dimensional_exponents** *contained some potentially ambiguous unqualified attributes. Remove the current EXPRESS specification and replace with:*

EXPRESS specification:

```
*)
FUNCTION derive_dimensional_exponents (x : unit):dimensional_exponents;
 LOCAL
   result : dimensional_exponents :=
            dimensional_exponents(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
 END_LOCAL;

 IF 'MEASURE_SCHEMA.DERIVED_UNIT' IN TYPEOF(x) THEN
  REPEAT i := LOINDEX(x\derived_unit.elements)
                        TO HIINDEX(x\derived_unit.elements);
    result.length_exponent := result.length_exponent +
    (x\derived_unit.elements[i]\derived_unit_element.exponent *
    x\derived_unit.elements[i]\derived_unit_element.unit
    \named_unit.dimensions.length_exponent);
    result.mass_exponent := result.mass_exponent +
    (x\derived_unit.elements[i]\derived_unit_element.exponent *
    x\derived_unit.elements[i]\derived_unit_element.unit
    \named_unit.dimensions.mass_exponent);
    result.time_exponent := result.time_exponent +
    (x\derived_unit.elements[i]\derived_unit_element.exponent *
    x\derived_unit.elements[i]\derived_unit_element.unit
    \named_unit.dimensions.time_exponent);
    result.electric_current_exponent := result.electric_current_exponent +
    (x\derived_unit.elements[i]\derived_unit_element.exponent *
    x\derived_unit.elements[i]\derived_unit_element.unit
    \named_unit.dimensions.electric_current_exponent);
    result.thermodynamic_temperature_exponent :=
    result.thermodynamic_temperature_exponent +
    (x\derived_unit.elements[i]\derived_unit_element.exponent *
    x\derived_unit.elements[i]\derived_unit_element.unit
    \named_unit.dimensions.thermodynamic_temperature_exponent);
    result.amount_of_substance_exponent :=
    result.amount_of_substance_exponent +
    (x\derived_unit.elements[i]\derived_unit_element.exponent *
     x\derived_unit.elements[i]\derived_unit_element.unit
     \named_unit.dimensions.amount_of_substance_exponent);
     result.luminous_intensity_exponent :=
     result.luminous_intensity_exponent +
     (x\derived_unit.elements[i]\derived_unit_element.exponent *
     x\derived_unit.elements[i]\derived_unit_element.unit
     \named_unit.dimensions.luminous_intensity_exponent);
   END_REPEAT;
   ELSE
      result := x\named_unit.dimensions;
   END_IF;
   RETURN (result);
 END_FUNCTION;
(*
```


Page 225, 21.5.3 valid_units

With the changes made in this technical corrigendum the **valid_units** *function requires some additions. Remove the current EXPRESS specification and replace with:*

EXPRESS specification:

```
*)
FUNCTION valid_units (m : measure_with_unit):BOOLEAN;
    IF 'MEASURE_SCHEMA.LENGTH_MEASURE' IN TYPEOF(m.value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'MEASURE_SCHEMA.MASS_MEASURE' IN TYPEOF(m.value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'MEASURE_SCHEMA.TIME_MEASURE' IN TYPEOF(m.value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'MEASURE_SCHEMA.ELECTRIC_CURRENT_MEASURE' IN
                              TYPEOF(m.value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
          dimensional_exponents(0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'MEASURE_SCHEMA.THERMODYNAMIC_TEMPERATURE_MEASURE' IN
          TYPEOF(m.value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'MEASURE_SCHEMA.CELSIUS_TEMPERATURE_MEASURE' IN
            TYPEOF(m.value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
          dimensional_exponents(0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
```



```
IF 'MEASURE_SCHEMA.AMOUNT_OF_SUBSTANCE_MEASURE' IN
      TYPEOF(m.value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
IF 'MEASURE_SCHEMA.LUMINOUS_INTENSITY_MEASURE' IN
                          TYPEOF(m.value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
       dimensional_exponents(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
IF 'MEASURE_SCHEMA.PLANE_ANGLE_MEASURE' IN TYPEOF(m.value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
     dimensional_exponents(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
IF 'MEASURE_SCHEMA.SOLID_ANGLE_MEASURE' IN TYPEOF(m.value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
     dimensional_exponents(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
IF 'MEASURE_SCHEMA.AREA_MEASURE' IN TYPEOF(m.value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
IF 'MEASURE_SCHEMA.VOLUME_MEASURE' IN TYPEOF(m.value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(3.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
IF 'MEASURE_SCHEMA.RATIO_MEASURE' IN TYPEOF(m.value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
IF 'MEASURE_SCHEMA.POSITIVE_LENGTH_MEASURE' IN
              TYPEOF(m.value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
IF 'MEASURE_SCHEMA.POSITIVE_PLANE_ANGLE_MEASURE' IN
```



```
              TYPEOF(m.value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
IF 'MEASURE_SCHEMA.ABSORBED_DOSE_MEASURE' IN TYPEOF(m.value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
    dimensional_exponents(2.0, 0.0, - 2.0, 0.0, 0.0, 0.0, 0.0) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
IF 'MEASURE_SCHEMA.ACCELERATION_MEASURE' IN TYPEOF(m.value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
    dimensional_exponents( 1.0, 0.0, -2.0, 0.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
IF 'MEASURE_SCHEMA.CAPACITANCE_MEASURE' IN
        TYPEOF(m.value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
    dimensional_exponents( -2.0, -1.0, 4.0, 1.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
IF 'MEASURE_SCHEMA.DOSE_EQUIVALENT_MEASURE' IN TYPEOF(m.value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
    dimensional_exponents(2.0, 0.0, - 2.0, 0.0, 0.0, 0.0, 0.0) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
IF 'MEASURE_SCHEMA.ELECTRIC_CHARGE_MEASURE' IN TYPEOF(m.value_component) THEN
  IF derive_dimensional_exponents(m.unit_component) <>
    dimensional_exponents( 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
    IF 'MEASURE_SCHEMA.CONDUCTANCE_MEASURE' IN
            TYPEOF(m.value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents( -2.0, -1.0, 3.0, 2.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
      END_IF;
  END_IF;
  IF 'MEASURE_SCHEMA.ELECTRIC_POTENTIAL_MEASURE' IN
          TYPEOF(m.value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents( 2.0, 1.0, -3.0, -1.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
      END_IF;
  END_IF;
  IF 'MEASURE_SCHEMA.ENERGY_MEASURE' IN TYPEOF(m.value_component) THEN
```



```
    IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents( 2.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0 ) THEN
      RETURN (FALSE);
    END_IF;
  END_IF;
  IF 'MEASURE_SCHEMA.FORCE_MEASURE' IN TYPEOF(m.value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents( 1.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0 ) THEN
      RETURN (FALSE);
    END_IF;
  END_IF;
  IF 'MEASURE_SCHEMA.FREQUENCY_MEASURE' IN TYPEOF(m.value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents( 0.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0 ) THEN
      RETURN (FALSE);
    END_IF;
  END_IF;
  IF 'MEASURE_SCHEMA.ILLUMINANCE_MEASURE' IN TYPEOF(m.value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents( -2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0 ) THEN
      RETURN (FALSE);
    END_IF;
  END_IF;
   IF 'MEASURE_SCHEMA.INDUCTANCE_MEASURE' IN TYPEOF(m.value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents( 2.0, 1.0, -2.0, -2.0, 0.0, 0.0, 0.0 ) THEN
      RETURN (FALSE);
    END_IF;
  END_IF;
  IF 'MEASURE_SCHEMA.LUMINOUS_FLUX_MEASURE' IN TYPEOF(m.value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0 ) THEN
      RETURN (FALSE);
    END_IF;
  END_IF;
  IF 'MEASURE_SCHEMA.MAGNETIC_FLUX_DENSITY_MEASURE' IN
                              TYPEOF(m.value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents( 0.0, -2.0, -1.0, -1.0, 0.0, 0.0, 0.0 ) THEN
      RETURN (FALSE);
  END_IF;
  END_IF;
   IF 'MEASURE_SCHEMA.MAGNETIC_FLUX_MEASURE' IN
                       TYPEOF(m.value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents( 2.0, 1.0, -2.0, -1.0, 0.0, 0.0, 0.0 ) THEN
      RETURN (FALSE);
    END_IF;
  END_IF;
  IF 'MEASURE_SCHEMA.POWER_MEASURE' IN TYPEOF(m.value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
      dimensional_exponents( 2.0, 1.0, -3.0, 0.0, 0.0, 0.0, 0.0 ) THEN
```



```
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'MEASURE_SCHEMA.PRESSURE_MEASURE' IN TYPEOF(m.value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents( -1.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
        IF 'MEASURE_SCHEMA.RADIOACTIVITY_MEASURE' IN
                          TYPEOF(m.value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0.0, 0.0, - 1.0, 0.0, 0.0, 0.0, 0.0) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'MEASURE_SCHEMA.RESISTANCE_MEASURE' IN TYPEOF(m.value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents( 2.0, 1.0, -3.0, -2.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    IF 'MEASURE_SCHEMA.VELOCITY_MEASURE' IN TYPEOF(m.value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents( 1.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
      END_IF;
    END_IF;
    RETURN (TRUE);
  END_FUNCTION;
(*
```

Page 253, Annex A

With the changes identified in this Technical Corrigendum the table of short names now contains extra entries. Remove the existing Annex A and replace with the following.

Annex A
(normative)

Short names of entities

Table A.1 provides the short names of entities specified in this part of ISO 10303. Requirements on the use of the short names are found in the implementation methods included in ISO 10303.

NOTE 1 The EXPRESS entity names are available from Internet:

<http://www.tc184-sc4.org/Short_Names/>

Table A.1 – Short names of entities

Entity names	Short names
ABSORBED_DOSE_MEASURE_WITH_UNIT	ADMWU
ABSORBED_DOSE_UNIT	ABDSUN
ACCELERATION_MEASURE_WITH_UNIT	AMW0
ACCELERATION_UNIT	ACCUNT
ACTION	ACTION
ACTION_ASSIGNMENT	ACTASS
ACTION_DIRECTIVE	ACTDRC
ACTION_METHOD	ACTMTH
ACTION_METHOD_ASSIGNMENT	ACMTAS
ACTION_METHOD_RELATIONSHIP	ACMTRL
ACTION_METHOD_ROLE	ACM0
ACTION_RELATIONSHIP	ACTRLT
ACTION_REQUEST_ASSIGNMENT	ACRQAS
ACTION_REQUEST_SOLUTION	ACRQSL
ACTION_REQUEST_STATUS	ACRQST
ACTION_RESOURCE	ACTRSR
ACTION_RESOURCE_RELATIONSHIP	ACRSRL
ACTION_RESOURCE_TYPE	ACRSTY
ACTION_STATUS	ACTSTT
ADDRESS	ADDRSS
AMOUNT_OF_SUBSTANCE_MEASURE_WITH_UNIT	AOSMWU
AMOUNT_OF_SUBSTANCE_UNIT	AOSU
APPLICATION_CONTEXT	APPCNT
APPLICATION_CONTEXT_ELEMENT	APCNEL
APPLICATION_CONTEXT_RELATIONSHIP	APCNRL
APPLICATION_PROTOCOL_DEFINITION	APPRDF
APPROVAL	APPRVL
APPROVAL_ASSIGNMENT	APPASS

Table A.1 – (continued)

Entity names	Short names
APPROVAL_DATE_TIME	APDTTM
APPROVAL_PERSON_ORGANIZATION	APPROR
APPROVAL_RELATIONSHIP	APPRLT
APPROVAL_ROLE	APPRL
APPROVAL_STATUS	APPSTT
AREA_UNIT	ARUNT
ATTRIBUTE_CLASSIFICATION_ASSIGNMENT	ATCLAS
ATTRIBUTE_VALUE_ASSIGNMENT	ATVLAS
ATTRIBUTE_VALUE_ROLE	ATVLRL
CALENDAR_DATE	CLNDT
CAPACITANCE_MEASURE_WITH_UNIT	CMWU
CAPACITANCE_UNIT	CPCUNT
CELSIUS_TEMPERATURE_MEASURE_WITH_UNIT	CTMWU
CERTIFICATION	CRTFCT
CERTIFICATION_ASSIGNMENT	CRTASS
CERTIFICATION_TYPE	CRTTYP
CHARACTERIZED_OBJECT	CHROBJ
CHARACTERIZED_OBJECT_RELATIONSHIP	CHOBRL
CLASSIFICATION_ASSIGNMENT	CLSASS
CLASSIFICATION_ROLE	CLSRL
CONDUCTANCE_MEASURE_WITH_UNIT	CMW0
CONDUCTANCE_UNIT	CNDUNT
CONTEXT_DEPENDENT_SHAPE_REPRESENTATION	CDSR
CONTEXT_DEPENDENT_UNIT	CNDPUN
CONTRACT	CNTRCT
CONTRACT_ASSIGNMENT	CNTASS
CONTRACT_RELATIONSHIP	CNTRLT
CONTRACT_TYPE	CNTTYP

Table A.1 – (continued)

Entity names	Short names
CONVERSION_BASED_UNIT	CNBSUN
COORDINATED_UNIVERSAL_TIME_OFFSET	CUTO
DATE	DATE
DATED_EFFECTIVITY	DTDEFF
DATE_AND_TIME	DTANTM
DATE_AND_TIME_ASSIGNMENT	DATA
DATE_ASSIGNMENT	DTASS
DATE_ROLE	DTRL
DATE_TIME_ROLE	DTTMRL
DERIVED_UNIT	DRVUNT
DERIVED_UNIT_ELEMENT	DRUNEL
DESCRIPTION_ATTRIBUTE	DSCATT
DIMENSIONAL_EXPONENTS	DMNEXP
DIRECTED_ACTION	DRCACT
DOCUMENT	DCMNT
DOCUMENT_PRODUCT_ASSOCIATION	DCP1
DOCUMENT_REFERENCE	DCMRFR
DOCUMENT_RELATIONSHIP	DCMRLT
DOCUMENT_REPRESENTATION_TYPE	DCRPTY
DOCUMENT_TYPE	DCMTYP
DOCUMENT_USAGE_CONSTRAINT	DCUSCN
DOCUMENT_USAGE_CONSTRAINT_ASSIGNMENT	DUCA
DOCUMENT_USAGE_ROLE	DCUSRL
DOCUMENT_WITH_CLASS	DCWTCL
DOSE_EQUIVALENT_MEASURE_WITH_UNIT	DEMWU
DOSE_EQUIVALENT_UNIT	DSEQUN
EFFECTIVITY	EFFCTV

Table A.1 – (continued)

Entity names	Short names
EFFECTIVITY_ASSIGNMENT	EFFASS
EFFECTIVITY_CONTEXT_ASSIGNMENT	EFC0
EFFECTIVITY_CONTEXT_ROLE	EFCNRL
EFFECTIVITY_RELATIONSHIP	EFFRLT
ELECTRIC_CHARGE_MEASURE_WITH_UNIT	ECM0
ELECTRIC_CHARGE_UNIT	ELCHUN
ELECTRIC_CURRENT_MEASURE_WITH_UNIT	ECMWU
ELECTRIC_CURRENT_UNIT	ELCRUN
ELECTRIC_POTENTIAL_MEASURE_WITH_UNIT	EPMWU
ELECTRIC_POTENTIAL_UNIT	ELPTU
ENERGY_MEASURE_WITH_UNIT	EMWU
ENERGY_UNIT	ENRUNT
EVENT_OCCURRENCE	EVNOCC
EVENT_OCCURRENCE_ASSIGNMENT	EVOCAS
EVENT_OCCURRENCE_CONTEXT_ASSIGNMENT	EOCA
EVENT_OCCURRENCE_CONTEXT_ROLE	EOCR
EVENT_OCCURRENCE_RELATIONSHIP	EVO0
EVENT_OCCURRENCE_ROLE	EVOCRL
EXECUTED_ACTION	EXCACT
EXPERIENCE	EXPRNC
EXPERIENCE_ASSIGNMENT	EXPASS
EXPERIENCE_RELATIONSHIP	EXPRLT
EXPERIENCE_ROLE	EXPRL
EXPERIENCE_TYPE	EXPTYP
EXPERIENCE_TYPE_ASSIGNMENT	EXTYAS
EXPERIENCE_TYPE_RELATIONSHIP	EXT0
EXPERIENCE_TYPE_ROLE	EXTYRL

Table A.1 – (continued)

Entity names	Short names
EXTERNALLY_DEFINED_ITEM	EXDFIT
EXTERNALLY_DEFINED_ITEM_RELATIONSHIP	EDIR
EXTERNAL_IDENTIFICATION_ASSIGNMENT	EXIDAS
EXTERNAL_REFERENT_ASSIGNMENT	EXRFAS
EXTERNAL_SOURCE	EXTSRC
EXTERNAL_SOURCE_RELATIONSHIP	EXSRRL
FORCE_MEASURE_WITH_UNIT	FMWU
FORCE_UNIT	FRCUNT
FREQUENCY_MEASURE_WITH_UNIT	FMW0
FREQUENCY_UNIT	FRQUNT
GENERAL_PROPERTY	GNRPRP
GENERAL_PROPERTY_ASSOCIATION	GNPRAS
GENERAL_PROPERTY_RELATIONSHIP	GNPRRL
GLOBAL_UNIT_ASSIGNED_CONTEXT	GUAC
GROUP	GROUP
GROUP_ASSIGNMENT	GRPASS
GROUP_RELATIONSHIP	GRPRLT
IDENTIFICATION_ASSIGNMENT	IDNASS
IDENTIFICATION_ASSIGNMENT_RELATIONSHIP	IDASRL
IDENTIFICATION_ROLE	IDNRL
ID_ATTRIBUTE	IDATT
ILLUMINANCE_MEASURE_WITH_UNIT	IMWU
ILLUMINANCE_UNIT	ILLUNT
INDUCTANCE_MEASURE_WITH_UNIT	IMW0
INDUCTANCE_UNIT	INDUNT
ITEM_IDENTIFIED_REPRESENTATION_USAGE	IIRU
LENGTH_MEASURE_WITH_UNIT	LMWU
LENGTH_UNIT	LNGUNT

Table A.1 – (continued)

Entity names	Short names
LIBRARY_ASSIGNMENT	LBRASS
LIBRARY_CONTEXT	LBRCNT
LOCAL_TIME	LCLTM
LOCATION	LCTN
LOCATION_ASSIGNMENT	LCTASS
LOCATION_RELATIONSHIP	LCTRLT
LOCATION_REPRESENTATION_ASSIGNMENT	LCRPAS
LOCATION_REPRESENTATION_ROLE	LCRPRL
LOCATION_ROLE	LCTRL
LOT_EFFECTIVITY	LTEFF
LUMINOUS_FLUX_MEASURE_WITH_UNIT	LFMWU
LUMINOUS_FLUX_UNIT	LMFLUN
LUMINOUS_INTENSITY_MEASURE_WITH_UNIT	LIMWU
LUMINOUS_INTENSITY_UNIT	LMINUN
MAGNETIC_FLUX_DENSITY_MEASURE_WITH_UNIT	MMWU
MASS_UNIT	MSSUNT
MEASURE_WITH_UNIT	MSWTUN
NAMED_UNIT	NMDUNT
NAME_ASSIGNMENT	NMASS
NAME_ATTRIBUTE	NMATT
OBJECT_ROLE	OBJRL
ORDINAL_DATE	ORDDT
ORGANIZATION	ORGNZT
ORGANIZATIONAL_ADDRESS	ORGADD
ORGANIZATIONAL_PROJECT	ORGPRJ
ORGANIZATIONAL_PROJECT_ASSIGNMENT	ORPRAS
ORGANIZATIONAL_PROJECT_RELATIONSHIP	ORP0
ORGANIZATIONAL_PROJECT_ROLE	ORPRRL

Table A.1 – (continued)

Entity names	Short names
ORGANIZATION_ASSIGNMENT	ORGASS
ORGANIZATION_RELATIONSHIP	ORGRLT
ORGANIZATION_ROLE	ORGRL
ORGANIZATION_TYPE	ORGTYP
ORGANIZATION_TYPE_ASSIGNMENT	ORTYAS
ORGANIZATION_TYPE_RELATIONSHIP	ORT0
ORGANIZATION_TYPE_ROLE	ORTYRL
PERSON	PERSON
PERSONAL_ADDRESS	PRSADD
PERSON_AND_ORGANIZATION	PRANOR
PERSON_AND_ORGANIZATION_ASSIGNMENT	PAOA
PERSON_AND_ORGANIZATION_ROLE	PAOR
PERSON_ASSIGNMENT	PRSASS
PERSON_ROLE	PRSRL
PERSON_TYPE	PRSTYP
PERSON_TYPE_ASSIGNMENT	PRTYAS
PERSON_TYPE_DEFINITION	PRTYDF
PERSON_TYPE_DEFINITION_ASSIGNMENT	PTDA
PERSON_TYPE_DEFINITION_FORMATION	PTDF
PERSON_TYPE_DEFINITION_RELATIONSHIP	PTD0
PERSON_TYPE_DEFINITION_ROLE	PTDR
PERSON_TYPE_ROLE	PRTYRL
PLANE_ANGLE_MEASURE_WITH_UNIT	PAMWU
PLANE_ANGLE_UNIT	PLANUN
POSITION_IN_ORGANIZATION	PSINOR
POSITION_IN_ORGANIZATION_ASSIGNMENT	PIOA
POSITION_IN_ORGANIZATION_RELATIONSHIP	PIO0
POSITION_IN_ORGANIZATION_ROLE	PIOR