INTERNATIONAL STANDARD ISO 10303-41:2005
TECHNICAL CORRIGENDUM 2

Published 2010-12-01

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 2

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 2

Technical Corrigendum 2 to ISO 10303-41:2005 was prepared by Technical Committee ISO/TC 184, *Automation systems and integration*, Subcommittee SC 4, *Industrial data*.

The purpose of the modifications to the text of ISO 10303-41:2005 and ISO 10303-41:2005/Cor.1:2008 is to add support for representation of a hierarchically linked sequence of representations, so as to allow unambiguous reference to a representation_item in a lower level of the hierarchy and to correct a file incompatibility problem by changing a DERIVEd attribute into a local Rule in the measure schema, to correct an error in the EXPRESS definition for dimensions_for_si_unit and to correct an error in the EXPRESS definition for valid_units.

Modifications to the text of ISO 10303-41:2005

Page 4
Insert the following new subclause immediately after 3.1.

3.2 Terms defined in ISO 10303-44

For the purposes of this part of ISO 10303, the following terms and definitions given in ISO 10303-44 apply:

— child node;

— descendent node;

— directed acyclic graph (DAG);

— leaf node;

— link;

— node;

— parent node;

— root node;

— tree.

Page 4, 3.2
Renumber subclause 3.2 as 3.3.

Pages 5 and 6, 3.3
Renumber subclause 3.3 as 3.4.
Renumber definition 3.3.1 as 3.4.1, definition 3.3.2 as 3.4.2, definition 3.3.3 as 3.4.4, definition 3.3.4 as 3.4.7 and definition 3.3.5 as 3.4.8.
Add the following new definitions.

3.4.3
chain
path in which all nodes and links are distinct

3.4.5
degree of a node
number of links in a graph that reference that node

3.4.6
path
sequence of nodes in a graph with the property that from each of it's nodes there is a link to the next node in the sequence

NOTE The first and last nodes in the path are different. No nodes are referenced more than once by a link.

Page 6, 3.4
Renumber subclause 3.4 as 3.5.

Page 33
Insert the following new subclause immediately after 5.5.4.

5.5.5 categories_of_product

The **categories_of_product** function returns the set of product category names associated with the argument. The function evaluates the bag of instances of **product category** that are associated with the input **product** through an instance of **product_related_product_category**. The function returns the SET of STRING values stored in the **name** attribute of the instances of **product_category** in the bag.

EXPRESS specification

```
*)
FUNCTION categories_of_product (obj : product) :SET OF STRING;
LOCAL
  category_assignments: BAG OF product_category;
  categories: SET OF STRING:=[];
END_LOCAL;
category_assignments := USEDIN(obj, 'PRODUCT_DEFINITION_SCHEMA.' +
'PRODUCT_RELATED_PRODUCT_CATEGORY.PRODUCTS');
REPEAT i := LOINDEX(category_assignments) TO HIINDEX(category_assignments) BY 1;
  categories := categories + category_assignments[i].name;
END_REPEAT;
RETURN(categories);
END_FUNCTION;
(*
```

Argument definitions

obj: the **product** whose categories are searched.

Page 34
Renumber subclause 5.5.5 as 5.5.6.

Page 50, Clause 7
*The added type **chained_representation_link** requires additions to the EXPRESS declaration in Clause 7. Remove the current EXPRESS definition and replace with the following.*

EXPRESS specification

```
*)
SCHEMA product_property_representation_schema;
REFERENCE FROM basic_attribute_schema                   -- ISO 10303-41
  (description_attribute,
   get_description_value,
   get_name_value,
   name_attribute);
REFERENCE FROM material_property_definition_schema      -- ISO 10303-45
  (property_definition_relationship);
REFERENCE FROM product_definition_schema                -- ISO 10303-41
  (product_definition,
   product_definition_relationship);
```

```
REFERENCE FROM product_property_definition_schema          -- ISO 10303-41
  (characterized_definition,
   general_property,
   product_definition_shape,
   property_definition,
   shape_aspect,
   shape_aspect_relationship);
REFERENCE FROM representation_schema                       -- ISO 10303-43
  (mapped_item,
   representation,
   representation_context,
   representation_item,
   representation_map,
   representation_relationship,
   using_representations);
REFERENCE FROM support_resource_schema                     -- ISO 10303-41
  (bag_to_set,
   label,
   text);
(*
```

Page 51, 7.3
Insert the following new subclause at the start of 7.3 and renumber the existing subclause 7.3.1 as 7.3.2.

7.3.1 chained_representation_link

The **chained_representation_link** type allows for the designation of a **mapped_item**, a **representation_context**, or a **representation_relationship**.

EXPRESS specification

```
*)
TYPE chained_representation_link = SELECT
  (mapped_item,
   representation_context,
   representation_relationship);
END_TYPE; --chained_representation_link

(*
```

Page 53
*The added entity **chain_based_item_identified_representation_usage** is a representation of a chain of instances of the entity data type **representation** that allows an unambiguous reference to an instance of the entity data type **representation_item** in a leaf node of a possibly multi-rooted graph of instances of the entity data type **representation**. Insert the following new subclause immediately after 7.4.2.*

7.4.3 chain_based_item_identified_representation_usage

A **chain_based_item_identified_representation_usage** is an **item_identified_representation_usage** that represents a chain of representations in a graph of representations, where the undirected links in the graph can be instances of entity data types **representation_context**, **representation_relationship**, or **mapped_item**. The directed links are derived from the nodes and undirected links and consist solely of instances of the entity data type **representation_relationship.** The attribute rep_1 of the derived instances is directed towards the root, while the attribute rep_2 is directed towards the leaf.

EXPRESS specification

```
*)
ENTITY chain_based_item_identified_representation_usage
 SUBTYPE OF (item_identified_representation_usage);
        nodes : LIST [2:?] OF UNIQUE representation;
  undirected_link : LIST [1:?] OF UNIQUE chained_representation_link;
 DERIVE
        root : representation := nodes[1];
 SELF\item_identified_representation_usage.used_representation RENAMED leaf :
        representation := nodes[HIINDEX(nodes)];
  directed_link : LIST [1:?] OF representation_relationship := get_directed_link
(nodes, undirected_link );
WHERE
 WR1 : root :=: directed_link [1]\representation_relationship.rep_1;
 WR2 : leaf :=: directed_link [HIINDEX(undirected_link
)]\representation_relationship.rep_2;
 WR3 : SIZEOF(nodes)  :=: SIZEOF(undirected_link ) + 1;
 WR4 : (SIZEOF(QUERY(directed_link_element<* directed_link  |
       (root :=: directed_link_element\representation_relationship.rep_1))) +
       SIZEOF(QUERY(directed_link_element<* directed_link |
       (root :=: directed_link_element\representation_relationship.rep_2)))) = 1;
 WR5 : (SIZEOF(QUERY(directed_link_element<* directed_link  |
       (leaf :=: directed_link_element\representation_relationship.rep_1))) +
       SIZEOF(QUERY(directed_link_element<* directed_link  |
       (leaf :=: directed_link_element\representation_relationship.rep_2)))) = 1;
 WR6 : (SIZEOF(QUERY(directed_link_element<* directed_link  |
       (root :<>: directed_link_element\representation_relationship.rep_1))) +
       SIZEOF(QUERY(directed_link_element<* directed_link  |
       (root :<>: directed_link_element\representation_relationship.rep_2))) +
       SIZEOF(QUERY(directed_link_element<* directed_link  |
       (leaf :<>: directed_link_element\representation_relationship.rep_1))) +
       SIZEOF(QUERY(directed_link_element<* directed_link  |
      (leaf :<>: directed_link_element\representation_relationship.rep_2)))) = 2;
 WR7 : NOT('REPRESENTATION_SCHEMA.MAPPED_ITEM' IN TYPEOF(undirected_link [1])) OR
       (root IN using_representations(undirected_link [1]));
 WR8 : SIZEOF(undirected_link ) = SIZEOF(directed_link );
END_ENTITY; --chain_based_item_identified_representation_usage

(*
```

Attribute definitions

nodes: the list of **representations** in the chain.

undirected_link : the list of items that relate nodes in the chain.

NOTE The items in the undirected_link are placed correctly relative to the nodes in the chain but are not guaranteed to have a consistent direction along the chain.

root: the node that is the initial terminus of the chain.

leaf: the node that is the final terminus of the chain.

directed_link : the list of **representation_relationships** that are derived from the undirected_link and that are consistently directed from the root to the leaf.

Formal propositions:

WR1 : The root shall be the node referenced by the attribute rep_1 of the initial member of the directed_link.

WR2 : The leaf shall be the node referenced by the attribute rep_2 of the final member of the directed_link.

WR3 : The tree shall be connected.

WR4 : The degree of the root node shall be one.

WR5 : The degree of the leaf node shall be one.

WR6 : The degree of any node other than the root node or the leaf node shall be two.

WR7 : If the first element in undirected_link is a **mapped_item**, then it shall be used in the root **representation**;.

WR8 : The size of the undirected_link shall equal the size of the directed_link.

Pages 54 to 56
Renumber subclauses 7.4.3 to 7.4.6 as 7.4.4 to 7.4.7.

Page 58, 7.5.3 get_property_definition_representations
Insert the following new subclause immediately after 7.5.3.

7.5.4 get_directed_link

The **get_directed_link** function returns for any list of **representations** and associated list of **chained_representation_link** the resolved list of **representation_relationships** when each member of undirected_link references the correct members of nodes. The **get_directed_link** function returns UNKNOWN when the input does not satisfy the correctness requirement.

EXPRESS specification

```
*)
FUNCTION get_directed_link  ( nodes : LIST OF representation;
             undirected_link   : LIST OF chained_representation_link)
                               : LIST OF representation_relationship;
LOCAL
 directed_link  : LIST OF representation_relationship := [] ;
END_LOCAL;

REPEAT i := 1 to SIZEOF(undirected_link );
 CASE TRUE OF
('REPRESENTATION_SCHEMA.REPRESENTATION_CONTEXT' IN TYPEOF(undirected_link [i])) :
 BEGIN
 IF ((nodes[i]\representation.context_of_items  :=: undirected_link [i]) AND
    (nodes[i+1]\representation.context_of_items :=: undirected_link [i])) THEN
    INSERT(directed_link, representation_relationship('','',nodes[i],nodes[i+1]),
(i - 1));
 ELSE
  RETURN(?);
 END_IF;
END;

('REPRESENTATION_SCHEMA.REPRESENTATION_RELATIONSHIP' IN TYPEOF(undirected_link
[i])) :
 BEGIN
 IF (((nodes[i]  :=: undirected_link [i]\representation_relationship.rep_1) AND
    (nodes[i+1] :=: undirected_link [i]\representation_relationship.rep_2)) OR
    ((nodes[i]  :=: undirected_link [i]\representation_relationship.rep_2) AND
    (nodes[i+1] :=: undirected_link [i]\representation_relationship.rep_1))) THEN
    INSERT(directed_link, representation_relationship('','',nodes[i],nodes[i+1]),
(i - 1));
 ELSE
  RETURN(?);
 END_IF;
END;

('REPRESENTATION_SCHEMA.MAPPED_ITEM' IN TYPEOF(undirected_link [i])) :
 BEGIN
 IF ((nodes[i] IN using_representations(undirected_link [i])) AND
    (nodes[i+1] :=: undirected_link
[i]\mapped_item.mapping_source\representation_map.mapped_representation)) THEN
    INSERT(directed_link, representation_relationship('','',nodes[i],nodes[i+1]),
(i - 1));
 ELSE
  RETURN(?);
 END_IF;
END;

  OTHERWISE : RETURN(?);
 END_CASE;
END_REPEAT;
RETURN(directed_link );
END_FUNCTION; --get_directed_link

(*
```

<u>Argument definitions:</u>

nodes : (input) the list of instances of the entity data type **representations** to be used as a reference for constructing the directed_link.

undirected_link : (input) the list of instances of the entity data type **mapped_item, representation_context,** or **representation_relationship** that the function is comparing against the reference to establish the directed_link.

Page 208, 21.4.7 conversion_based__unit

The definition of this entity in ISO 10303-41:2005/Cor.1:2008 causes a file incompatibility problem in that the inherited attribute dimensions is required to be populated with an asterisk "" instead of an instance identifier. Post-processors conformant to previous editions of ISO 10303-41 may not be able to correctly interpret the asterisk.*
Replace the EXPRESS definition in ISO 10303-41:2005 and ISO 10303-41:2005/Cor.1:2008 with the following:

<u>EXPRESS specification</u>

```
*)
ENTITY conversion_based_unit
  SUBTYPE OF (named_unit);
  name : label;
  conversion_factor : measure_with_unit;
WHERE
WR1: SELF\named_unit.dimensions =
derive_dimensional_exponents(conversion_factor\measure_with_unit.unit_component);
END_ENTITY;
(*
```

Page 224, 21.5.2 dimensions_for_si_unit

The definition of the dimensional exponents for farad is incorrect. Remove the current EXPRESS definition and replace with the following:

EXPRESS specification

```
*)
FUNCTION dimensions_for_si_unit (n : si_unit_name) : dimensional_exponents;
 CASE n OF
  metre :      RETURN(dimensional_exponents (1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
  gram :        RETURN(dimensional_exponents(0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0));
  second :      RETURN(dimensional_exponents(0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0));
  ampere :      RETURN(dimensional_exponents(0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0));
  kelvin :      RETURN(dimensional_exponents(0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0));
  mole :        RETURN(dimensional_exponents(0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0));
  candela :     RETURN(dimensional_exponents(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
  radian :      RETURN(dimensional_exponents(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
  steradian :   RETURN(dimensional_exponents(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
  hertz :      RETURN(dimensional_exponents(0.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0));
  newton :     RETURN(dimensional_exponents(1.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
  pascal :    RETURN(dimensional_exponents(-1.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
  joule :      RETURN(dimensional_exponents(2.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
  watt :       RETURN(dimensional_exponents(2.0, 1.0, -3.0, 0.0, 0.0, 0.0, 0.0));
  coulomb :     RETURN(dimensional_exponents(0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0));
  volt :      RETURN(dimensional_exponents(2.0, 1.0, -3.0, -1.0, 0.0, 0.0, 0.0));
  farad :     RETURN(dimensional_exponents(-2.0, -1.0, 4.0, 2.0, 0.0, 0.0, 0.0));
  ohm :       RETURN(dimensional_exponents(2.0, 1.0, -3.0, -2.0, 0.0, 0.0, 0.0));
  siemens :   RETURN(dimensional_exponents(-2.0, -1.0, 3.0, 2.0, 0.0, 0.0, 0.0));
  weber :     RETURN(dimensional_exponents(2.0, 1.0, -2.0, -1.0, 0.0, 0.0, 0.0));
  tesla :     RETURN(dimensional_exponents(0.0, 1.0, -2.0, -1.0, 0.0, 0.0, 0.0));
  henry :     RETURN(dimensional_exponents(2.0, 1.0, -2.0, -2.0, 0.0, 0.0, 0.0));
  degree_Celsius :
                RETURN(dimensional_exponents(0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0));
  lumen :       RETURN(dimensional_exponents(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
  lux :        RETURN(dimensional_exponents(-2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
  becquerel :  RETURN(dimensional_exponents(0.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0));
  gray :       RETURN(dimensional_exponents(2.0, 0.0, -2.0, 0.0, 0.0, 0.0, 0.0));
  sievert :    RETURN(dimensional_exponents(2.0, 0.0, -2.0, 0.0, 0.0, 0.0, 0.0));
  OTHERWISE :  RETURN(?);
 END_CASE;
END_FUNCTION; -- dimensions_for_si_unit
(*
```

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 the following:

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 RETURNR(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, 2.0, 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;
```