Recommended Practices

for

Persistent IDs for Design Iteration and Downstream Exchange

Release 1.00

November 28, 2023

CAx-IF		
Jochen Boy PROSTEP AG jochen.boy@prostep.com	**Robert Lipman** NIST robert.lipman@nist.gov	**Phil Rosché** ACCR, LLC. phil.rosche@accr-llc.com

Technical	
Asa Trainer Consultant agtrainer@comcast.net	Thomas Thurman Consultant thomas.r.thurman@imonmail.com

Table of Contents

List of Figures

Document History

This document is a new CAx-IF Recommended Practice and adds new constructs.

Release	Date	Change
0.1	2018-09-26	Initial release
0.2	2019-09-05	Rewritten to support either Part 21 Data Section (Attribute) approach and Part 21 E3 Anchor Section approach
0.3	2019-12-27	Added instance diagrams; limited Anchor Section discussion to refer to a future edition of the document
0.4	2020-01-14	Editorial revision
0.5	2020-05-28	Updated instantiation diagrams (Figures 2, 4, 5, 6)
0.6	2022-12-29	Updated to include UUID_ATTRIBUTE schema, versioning requirements, and iterative loading of imported data with UUIDs to emphasize the design iteration use case rather than the downstream-use use case;
0.7	2023-01-25	Additional tweaks before pre-release reviews including updated diagrams (Figures 2 through 7). Draft 0.7 out for review.
0.8	2023-02-01	Updated Mikael’s information; Draft 0.8 out for review
0.9	2023-02-03	Minor changes to clarify use of UUID_attribute subtypes rather than abstract UUID_attribute supertype; Draft 0.9 out for CAx review
0.91	2023-05-18	Replaced all references to “GUID” with “UUID”. This is based on ad hoc agreement with DMSC who are replacing “QPID” with “UUID” as well in QIF v4.0 (confirm this version with Larry or Curtis). Add highlights to the subtypes in section 4.2.2 to identify those subtypes that will limit scope and be the focus for this Rec Prac and the R52J and later Test Cases that reference it. Allow the publishing of UUIDs in either the Data Section form or the Anchor Section form. Either form is valid and post-processors should be able to read either type.
0.92	2023-06-07	Updated schema to push UUID_ATTRIBUTE under the Merkle tree; see Sections 4.2.2 and 5. Initial instantiation model is very similar to version 0.9 as the UUID_ATTRIBUTE inherits the ‘uuid’ identifier from uuid_leaf_node that has no other mandatory attributes.
0.93	2023-06-08	Replaced Figures 2 and 3. Changed P21 snippet in clause 4.2.1 to Example 1. Corrected record #1 in Example 1. Added notes to uuid_leaf_node, uuid_root_node, uuid_internal_node.
0.94	2023-06-28	Replaced Figures 2 through 8. Changed p21 snippet in Example 1. Updated EXPRESS in Clause 4.2.2: uuid_attribute_select, uuid_relationship_role,uuid_attribute, hash_based_v5_uuid_attribute, uuid_tree_node, uuid_leaf_node, uuid_internal_node, uuid_root_node, and uuid_context_role. Removed 4.4.2.1 id_attribute as we are no longer including id_attribute between uuid_attribute and the target. Updated Technical Authors info. Updated notes on Figures 4 through 8.Added link to trial schema in Annex A.
0.95	2023-11-20	Replaced Figures 2 and 3. Changed uuid_attribute.identified_item to: LIST [1:?] OF UNIQUE LIST[1:?] OF UNIQUE uuid_attribute_select. Note that Figure 2 includes a text note to this effect. Replaced uuid_leaf_node.data with a reference to uuid_attribute_select. Updated specification of file identification.
1.00	2023-11-28	All current changes accepted; DRAFT status changed to Released

Acknowledgements

The authors also wish to thank Mr. Mikael Hedlind, Sandvik, Sweden, for his input on requirements for data exchange between CAD systems for design iteration and his development of sample test models for discussion on the design iteration exchange use case.

Portions of this report were prepared by Thomas Thurman dba TRThurman Consuling using Federal funds under Award No. 70NANB22H217 from the Natonal Ins6tute of Standards and Technology (NIST), U.S. Department of Commerce. The statements, findings, conclusions, and recommenda6ons are those of the author(s) and do not necessarily reflect the views of NIST or the U.S. Department of Commerce.

1 Introduction

Research into mechanisms for maintaining traceability of engineering product data during exchange has concluded that the introduction and tracking of persistent IDs in that product data are feasible. Achieving traceability of individual data elements during state changes in product design and product development process can provide immediate practical benefits for several use cases. In this Recommended Practices document, we will focus on two such use cases – Design Iteration and Downstream Exchange. Review and discussion by CAx-IF at the April 2019 meeting, concluded on limiting the testing scope to Downstream Exchange. Design Iteration will be covered in a future version. After limited participation for the existing Downstream-use use case in 2020 and 2021 test rounds, and increasing interest in the Design Iteration use case, a discussion at the September 2022 CAx-IF meeting concluded that focus should shift to the latter use case.

1.1 Design Iteration

A significant problem in design in terms of efficiency, is the difficulty in referencing shared data between team members (teaming partners or OEMs and their suppliers) during design iteration. Once a product model recipient consumes a source model into their CAD system and references product model entities from that source model in their design, they become locked into that instance of the external data. When subsequent versions of that source data are received and consumed again, re-mapping the references by hand, to maintain associativity between the now updated source data and the existing target design data, is challenging at best and nearly impossible for any but the most trivial of models.

Over the years, several major CAD vendors as well as at least one independent interoperability vendor, have implemented the ability to perform this associative mapping and update process automatically (Reference PTC's "Associative Topology Bus" (patent), NX's "Associative Update", Dassault Systemes' "Topological Naming", Integration Guard, etc.). These implementations have been successful, to a greater or lesser extent, using customized direct translation methods or within the boundaries of the individual vendor's exchange ecosystem. Implementation across system boundaries poses challenges including differences between systems in how and when entities are modified during design changes as well as differences in how each CAD system identifies model elements and the permanence of the identifiers for elements during change. For example, some systems may transform a geometric entity internally while maintaining its identifier while others may simply remove and replace the entity and update its own internal references on the fly. Another challenge is subtle differences between mathematical approaches to entity modeling in each CAD system and the impact those mathematical differences have on how entities are mapped. A well-known example is the modeling of hole features where some systems model the hole as a single cylindrical surface and other map the hole as two half cylinders. This example of a one-to-many surface mapping requires solid bookkeeping of entity identifiers by the receiving system.

In short, the ability to retain associativity between source feature, geometry, topology, and or attribute data and similar target data that may have dependencies on (references to) those source elements are key requirement for rapid iteration between product versions, thus improving design efficiency and reducing product development cost.

1.2 Downstream Exchange

Passing of design data to downstream systems has similar needs to maintain associative references between, for example, manufacturing tool paths and the model entities they are derived from, or dimensional tolerances as planned and measured in metrology systems and those same dimensional tolerances as defined in the original design systems where the models were created. In addition to the challenges of controlling and managing change in downstream systems based on model state change in designs, there is a second and perhaps more burdensome requirement for downstream consumption of model data and that is traceability. Traceability is necessary from the original system of creation of data through all its uses and throughout its lifecycle in order to follow potential faults to their source - either design, manufacture, material, or process. This traceability is a necessary and required forensic tool to be used in legal proceedings and technical investigations where complex product or process failures may have resulted in injury or death. In particular, the manufacturing and metrology communities have long had procedures in place to tag and maintain identification of products and their components, constituent entities, and attributes, to maintain this traceability and ensure their ability to isolate and identify any suspect system characteristics or entities in the event of failure. The ability to rapidly trace through suspect systems may be critical to quickly rectifying dangerous problems in product designs or product development processes.

1.3 Maintenance of this Document

This document will be maintained by the CAx-IF and will cover the Part 21 based implementations of persistent IDs. In the current version, this document will focus on the design iteration use case. In a future version, the document will return to refocus on the metrology use case.

2 Scope

The following are within the scope of this document:

- The generation and use of Universally Unique Identifiers (UUIDs) (see Section 4 below) for maintaining associativity of entities in iterative design.
- The generation and use of Universally Unique Identifiers (UUIDs) (see Section 4 below) for maintaining associativity of entities for traceability for downstream uses of the model.
- The use of Part 21 Data Section or Part 21 Ed 3 Anchor Section for assigning UUIDs either on initial publication of the STEP document or after the original publishing of the STEP document should be supported.

3 Document Identification

For validation purposes, STEP processors shall state which Recommended Practice document and version have been used in the creation of the STEP file. This will not only indicate what information a consumer can expect to find in the file, but even more important where to find it in the file.

This shall be done by adding a pre-defined ID string to the `description` attribute of the `file_description` entity in the STEP file header, which is a list of strings. The ID string consists of four values delimitated by a triple dash (‘---‘). The values are:

```
Document Type---Document Name---Document Version---Publication Date
```

The string corresponding to this version of this document is:

`CAx-IF Rec.Pracs.---Persistent IDs---1.00---2023-11-28`

It will appear in a STEP file as follows:

```
FILE_DESCRIPTION(('...','CAx-IF Rec.Pracs.---Persistent IDs---1.00---2023-
11-28',),'2;1');
```

4 Persistent IDs

A mechanism for generating uniformly consistent, cross-application entity IDs is required for the above processes to work. These IDs need to be unique to prevent clashes between entity identifiers. In the issue summary for ISO Jira Task BS10303-3834 (formerly Bugzilla #5901) written by Thomas Thurman (see also BRUTUS #23), the scope of such uniqueness was suggested as only being required within the context of a specific product. This might be considered sufficient if only the first use case – design iteration – was required. In the context of the second use case, however, particularly the metrology use case, persistent (permanent), universally unique identifiers are necessary and need to be applied to product (as well as product effectivity, i.e., serialized product artifacts, if they exist), and to individual semantic PMI entities in the product.

4.1 Formulation of Identifiers (UUIDs)

Fortunately, persistent, universally unique identifiers have been in use in the information technology domain for a long time. Such a 'universally unique identifier' (UUID) is a 128-bit number used to identify information in computer systems, e.g., operating systems, databases, and communications processes. The term 'globally unique identifier' (GUID) is sometimes also used. UUIDs have been standardized by the Open Software Foundation (OSF) and are documented as part of ISO/IEC 11578:1996 "Information technology – Open Systems Interconnection – Remote Procedure Call (RPC)" and more recently in ITU-T Rec. X.667 | ISO/IEC 9834-8:2005. Most computing platforms provide convenient support for generating them, and for parsing their textual representation. More detail about UUIDs can be found on Wikipedia. Within the engineering domain, such UUIDs or GUIDs are already in place and being used in the Industry Foundation Classes (IFC) format of the Building Information Model (BIM) and in the Quality Information Framework (QIF) standard for the Metrology domain. A recent agreement has been reached to standardize the nomenclature to UUID in this STEP recommended Practice as well as the latest release of DMSC's QIF v4.0 release.

In the above standard there are 5 possible versions of UUIDs:

- Version 1 UUIDs are generated from a time and a node id (usually the MAC address),
- Version 2 UUIDs are generated from an identifier (usually a group or user id), time, and a node id,
- Versions 3 and 5 produce deterministic UUIDs generated by hashing a namespace identifier and name,
- Version 4 UUIDs are generated using a random or pseudo-random number.

Currently, the QIF standard suggests the use of Version 4 UUIDs, i.e., via random number seed. Several tools are generating UUIDs using this UUID Version. Based on research, there is some desirability to be able to consistently reproduce a UUID from some namespace and name data. Versions 3 and 5 allow this but the algorithm of Version 3 has been deprecated in favor of Version 5. Though there has been some discussion of registering a namespace specifically for these engineering information exchange use cases, and the namespace hierarchy might include the name and version of the generating preprocessor, no definitive plans have been put in place. The namestring data for the seed to generate this UUID version might be a fully qualified path from the product identifier to the entity in question. See the example below.

A version 5 UUID could be constructed, for example, from:

- A pre-defined namespace: uuid.NAMESPACE_DNS
  - For example: UUID('6ba7b810-9dad-11d1-80b4-00c04fd430c8')
- Model Identification String (SHA-1 hash of string) composed of

  Filename: "nist_ftc_06_asme1_nx900_rd.prt"

Type: “PMI Feature Control Frame”

Persistent ID: “ID 879819”

The function **uuid.uuid5** with arguments **(uuid.NAMESPACE_DNS, "nist_ftc_06_asme1_nx900_rd.prt PMI Feature Control Frame - ID 879819")**

renders **UUID('491b0d21-fc8e-50d0-874f-6b7f5a95c47a')**.

It is important to note that, if all hash string elements are the same, the UUID generator will generate the same UUID again.

Model Identification String (SHA-1 Hash String) is the responsibility of the owner (creator) application. The creator application should concatenate string elements including the following:

- a unique part name or part number,
- a part instance serial number (if it exists),
- an entity type or full path from the product to the individual entity, and
- an internally maintained, unique persistent entity ID.

In the remainder of this recommended practice document, we will refer to the term UUID when describing these unique identifiers.

4.2 IDs in STEP

There are two valid approaches to storing UUIDs in STEP. The first is via an internal identifier within the context of the Part 21 data section. This method was originally suggested by data modelers and may make sense for the original publishers of STEP from the source CAx system and will be described in Section 4.2.2 below. The second is the storing of such identifiers not in the data section but rather within the Part 21 Edition 3 Anchor Section. This second method was once put forward primarily as an alternative method to add identifiers to a STEP file for newly appended data after its original publication or to allow addition of identifiers to legacy STEP data. Either of the above methods – Data Section or Anchor Section – are considered valid for initial publication by the source preprocessor or for appending data after initial publication.

4.2.1 Model Versioning to Support Iteration of Product Design

To support the design iteration use case, some mechanism is needed, at the application level, to identify the change of state of the model from one iteration to the next. This is typically done by checking the model out of a PLM system, performing some modification on the design model, and checking that next model iteration back into the PLM system. This requirement also applies between revisions of the model, i.e., when major changes to the design are published for use within the extended enterprise. In either case (iteration or revision), a counter is incremented to identify the change in state of the model. Often this counter information is carried not only within the PLM system, but also within the model itself. In order to support iterative design and update during STEP exchanges, this iteration flag, i.e., the revision/version counter, also needs to be injected into the STEP file when created and also needs to be captured by any consuming STEP postprocessor. The format of this iteration flag is a string in form of <revision letter>.<version number> or <revision letter>-<version number>, e.g. A.1, A.2, B.1, B.2, etc, or A-1, A-2, B-1, B-2, etc. The `product_definition` and `product_definition_formation` entities are used for this purpose. An example of the connection between these two entities and a `UUID_attribute` entity (Section 4.2.2 below) is given in example 1 below:

```
Example 1

#1=v5_uuid_attribute ('36 character uuid string',(#2));
/*identified_item is the second attribute in the attribute
list */

#2=product_definition('','',#3,#4);

#3=product_definition_formation('A.1','',#5);

#4=product_definition_context(...);

#5=product(....);
```

4.2.2 Persistent ID (UUID) Entity Identification

The EXPRESS entities and attributes used to support the requirements of UUID entity identification and relationships between them are illustrated below (Figure 1, also refer to the schema diagram in Figure 2). They are proposed to be included in AP242 Edition 4. In case of any discrepancies between these test schema entries and the published AP242 Edition 3 should be brought to the attention of the authors.

Note that the full list of types for `id_attribute_select` and `identification_item` are given below, **only those shown in bold** below will be used for the purposes of entity identification for persistent ID usage.

TYPE id_attribute_select = SELECT
(action,
address,
application_context,
ascribable_state_relationship,
dimensional_size, [**dimensional_size_with_path**]
geometric_tolerance,
group,
organizational_project,
product_category,
property_definition,
representation, [advanced_brep_shape_representation, **shape_representation**,]
shape_aspect,
shape_aspect_relationship, **[dimensional_location]**
topological_representation_item); [advanced_face, closed_shell, open_shell]
END_TYPE;

TYPE identification_item = SELECT
(
action,
application_context,
characterized_object,
characterized_object_relationship,
context_dependent_shape_representation,
derived_unit,
dimension_related_tolerance_zone_element,
dimensional_characteristic_representation,

```
    dimensional_location,
    founded_item,
    geometric_tolerance_auxiliary_classification,
    geometric_tolerance_relationship,
    gps_filter,
    gps_filtration_specification,
    invisibility,
    item_identified_representation_usage,
    limits_and_fits,
    measure_qualification,
    measure_with_unit,
    named_unit,
    plus_minus_tolerance,
    representation_item,
    representation_item_relationship,
    runout_zone_orientation,
    tolerance_value,
    tolerance_zone_definition,
    tolerance_zone_form,
       action_directive,
       action_directive_relationship,
       action_method,
       action_method_relationship,
       action_property,
       action_property_representation,
       action_relationship,
       address,
       alternate_product_relationship,
       alternative_solution_relationship,
       analysis_assignment,
       analysis_representation_context,
       applied_action_assignment,
       applied_action_method_assignment,
       applied_action_request_assignment,
       applied_approval_assignment,
       applied_certification_assignment,
       applied_classification_assignment_relationship,
       applied_contract_assignment,
       applied_description_text_assignment,
       applied_description_text_assignment_relationship,
       applied_document_reference,
       applied_document_usage_constraint_assignment,
       applied_effectivity_assignment,
       applied_event_occurrence_assignment,
       applied_external_identification_assignment,
       applied_external_identification_assignment_relationship,
       applied_identification_assignment,
       applied_ineffectivity_assignment,
       applied_organization_assignment,
       applied_organizational_project_assignment,
       applied_person_and_organization_assignment,
       applied_security_classification_assignment,
       applied_time_interval_assignment,
       applied_usage_right,
       approval,
       approval_relationship,
       approval_status,
       ascribable_state,
       ascribable_state_relationship,
       assembly_component_usage,
       assembly_component_usage_substitute,
       assignment_object_relationship,
       breakdown_element_realization,
       breakdown_of,
       certification,
       change_group,
       characterized_class,
       class,
       class_system,
       configuration_effectivity,
       configuration_item,
       configuration_item_relationship,
       contract,
       contract_relationship,
       date_and_time_assignment,
       date_assignment,
       degenerate_pcurve,
       dimensional_size,
       dimensional_size_with_path,
       directed_action_assignment,
       document_file,
       document_relationship,
       document_type,
       draughting_model,
       effectivity,
       effectivity_relationship,
       envelope,
       envelope_relationship,
       evaluated_characteristic,
       event_occurrence,
       event_occurrence_relationship,
       evidence,
       exclusive_product_concept_feature_category,
       executed_action,
       general_property,
       general_property_relationship,
       generic_property_relationship,
       group,
       group_relationship,
       identification_assignment_relationship,
       information_right,
       information_usage_right,
       interface_connection,
       interface_connector_as_planned,
       interface_connector_as_realized,
       interface_connector_definition,
       interface_connector_design,
       interface_connector_occurrence,
       interface_connector_version,
       interface_definition_connection,
       interface_definition_for,
       interface_specification_definition,
       interface_specification_version,
       link_motion_relationship,
       material_designation,
       material_designation_characterization,
       measure_representation_item,
       mechanical_design_geometric_presentation_representation,
       message_relationship,
       organization,
       organization_relationship,
       organizational_address,
       organizational_project,
       organizational_project_relationship,
       package_product_concept_feature,
       person,
       person_and_organization,
       person_and_organization_address,
       point_on_surface,
       presentation_area,
       process_operation,
       process_plan,
       product,
       product_category,
       product_class,
       product_concept,
       product_concept_context,
       product_concept_feature,
       product_concept_feature_category,
       product_concept_relationship,
       product_definition,
       product_definition_formation,
       product_definition_formation_relationship,
       product_definition_occurrence,
       product_definition_occurrence_reference,
       product_definition_relationship,
       product_definition_usage,
       product_definition_usage_relationship,
       product_group,
       product_group_membership,
       product_group_relationship,
       product_identification,
       product_process_plan,
       product_relationship,
       property_definition,
       property_definition_relationship,
       property_definition_representation,
       representation,
       representation_context,
       representation_relationship,
       requirement_assignment,
       requirement_for_action_resource,
       requirement_source,
       retention,
       rule_set,
       satisfies_requirement,
       security_classification,
       security_classification_level,
       shape_aspect,
       shape_aspect_relationship,
       shape_feature_definition,
       shape_feature_definition_relationship,
       shape_representation,
       state_definition_to_state_assignment_relationship,
       state_observed,
       state_observed_assignment,
       state_observed_relationship,
       state_type,
       state_type_assignment,
       state_type_relationship,
       structured_message,
       time_interval,
       time_interval_relationship,
       usage_association,
       validation,
       verification,
       verification_relationship,
       versioned_action_request,
       versioned_action_request_relationship);
   END_TYPE;


   ENTITY id_attribute;
     attribute_value : identifier;
     identified_item : id_attribute_select;
   END_ENTITY;


     TYPE uuid_attribute_select = SELECT
       (id_attribute_select,
        identification_item);
     END_TYPE;

     TYPE uuid = STRING (36) FIXED;
     END_TYPE;

     TYPE uuid_relationship_role = ENUMERATION OF
       (SUPERSEDES,
        MERGE,
        SPLIT,
        DERIVE_FROM,
        SAME_AS,
        SIMILAR_TO);
     END_TYPE;


   ENTITY uuid_attribute
     ABSTRACT SUPERTYPE OF(ONEOF(
       v5_uuid_attribute,
       v4_uuid_attribute)
       ANDOR uuid_attribute_with_approximate_geometric_location)
      identifier : uuid;
     identified_item : LIST [1 : ?] OF UNIQUE LIST [1:?] OF UNIQUE
   uuid_attribute_select;
    UNIQUE
     UR1 : identifier;
   END_ENTITY;

     ENTITY v5_uuid_attribute
       SUBTYPE OF(uuid_attribute);
     END_ENTITY;

     ENTITY v4_uuid_attribute
       SUBTYPE OF(uuid_attribute);
     END_ENTITY;
     ENTITY hash_based_v5_uuid_attribute
       SUBTYPE OF(v5_uuid_attribute);
         hash_function      : STRING;
       WHERE
         WR1 : hash_function <> '';
     END_ENTITY;

     ENTITY uuid_attribute_with_approximate_location
       SUBTYPE OF(uuid_attribute);
         location_representation : shape_representation;
         approximate_location    : cartesian_point;
       WHERE
         WR1 : location_representation IN
   using_representations(approximate_location);
     END_ENTITY;

   *)
     ENTITY uuid_relationship;
         identifier : uuid;
         uuid_1     : uuid;
         uuid_2     : uuid;
         role       : uuid_relationship_role;
         tree_root  : OPTIONAL uuid_tree_root;
       UNIQUE
         UR1 : identifier;
       WHERE
         WR1 : uuid_1 <> uuid_2;;
         WR2 : uuid_1 <> identifier;;
```

```
        WR3 : identifier <> uuid_2;;
        wr4 : NOT ((parent_child) = role) OR EXISTS(tree_root);
    END_ENTITY;

    ENTITY uuid_provenance;
        identifier : uuid;
        content    : LIST [1:?] OF UNIQUE uuid_relationship;
      UNIQUE
        UR1 : identifier;
    END_ENTITY;

    ENTITY uuid_tree_node
      ABSTRACT
      SUPERTYPE OF (ONEOF(uuid_leaf_node, uuid_internal_node));
        identifier : uuid;
        node_2     : OPTIONAL uuid_tree_node;
        node_1     : OPTIONAL uuid_tree_node;
      WHERE
        WR1 : node_1 <> node_2;
    END_ENTITY;

  ENTITY uuid_leaf_node
      SUBTYPE OF(uuid_tree_node);
     data : uuid_attribute_select;
      DERIVE
        leaf_operand : STRING (1) FIXED := '0';
      WHERE
        WR1 : NOT (EXISTS (node_1) OR EXISTS(node_2));
        WR2 :
  (SIZEOF(USEDIN(SELF,'AP242_MANAGED_MODEL_BASED_3D_ENGINEERING_MIM_LF.
  UUID_TREE_NODE.NODE_1')) = 1) AND
  (SIZEOF(USEDIN(SELF,'AP242_MANAGED_MODEL_BASED_3D_ENGINEERING_MIM_LF.
  UUID_TREE_NODE.NODE_2')) = 1);  END_ENTITY;

    ENTITY uuid_internal_node
      SUBTYPE OF(uuid_tree_node);
      DERIVE
        internal_operand : STRING (1) FIXED := '1';
      WHERE
        WR1 : EXISTS(node_1) AND EXISTS(node_2);
        WR2 : (SIZEOF(USEDIN(SELF,
  'AP242_MANAGED_MODEL_BASED_3D_ENGINEERING_MIM_LF.UUID_TREE_NODE.NODE_
  1'))=1) AND (SIZEOF(USEDIN(SELF,
  'AP242_MANAGED_MODEL_BASED_3D_ENGINEERING_MIM_LF.UUID_TREE_NODE.NODE_
  2')) = 1);
    END_ENTITY;

    ENTITY uuid_root_node
      SUBTYPE OF(uuid_internal_node);
        hash_function : STRING;
      DERIVE
        root_operand  : STRING := '1';
    WHERE
        WR1 : SIZEOF(USED_IN(UUID_SCHEMA.UUID.TREE_NODE.NODE_1)) = 0;
        WR2 : SIZEOF(USED_IN(UUID_SCHEMA.UUID.TREE_NODE.NODE_2)) = 0;
        WR3 : EXISTS(SELF\uuid_tree_node.node_1) AND
EXISTS(SELF\uuid_tree_node.node_2);
  END_ENTITY;
(
  ENTITY uuid_context_role;
        identifier : uuid;
        role       : STRING;    UNIQUE      UR1 : identifier;
WHERE
    WR1 : role <> '';  END_ENTITY;
```

Figure 1: EXPRESS Entities for Persistent IDs

4.2.2.1 UUID_ATTRIBUTE

The uuid_attribute entity is an extension from the id_attribute entity that represents UUID-specific identifier information. The uuid_attribute is abstract and only subtype of uuid_attribute must be populated, i.e. v5_uuid_attribute and v4_uuid_attribute, as appropriate. A uuid_attribute associates a UUID with an ordered collection of product data items. Only items specified by id_attribute_select, or those specified by identification_item, shall be specified by uuid_attribute. . In the first trial, the ordered collection will contain one item.

As a follow-on trial, the path from shape_aspect to gisu to advanced_face will be exchanged.

As a further follow-on trial, the merkle tree (shape_aspect…advanced_face, shape_aspect…geometric_tolerance, shape_aspect…presentation geometry) will be exchanged.

4.2.2.2 V5_UUID_ATTRIBUTE

The v5_uuid_attribute entity is a SUBTYPE OF uuid_attribute, and inherits the attributes from that ENTITY. A v5_uuid_attribute is a uuid_attribute that provides a UUID that conforms to version 5 of the relevant rfc. Version 5 UUIDs are generated based on a known namespace identifier and a name string that can be relied on between iterations of the product. The names string nor the namespace are provided. They are left to the implementor to manage internally.

4.2.2.3 HASH_BASED_V5_UUID_ATTRIBUTE

A hash_based_v5_uuid_attribute entity is a v5_uuid_attribute that provides a UUID that is a hash of the data items specified in the identified_items list attribute in the SUPERTYPE uuid_attribute. The hash function name shall be provided, but may be an empty string.

4.2.2.4 V4_UUID_ATTRIBUTE

The v4_uuid_attribute entity represents UUID identifier information. This entity collects the value of one of the UUID subtypes and the STEP entity that that UUID is assigned to. A `uuid_attribute` associates a UUID with an ordered collection of product data items. A `v4_uuid_attribute` is a `uuid_attribute` that provides a UUID that conforms to version 4 of the relevant rfc. Version 4 UUIDs are generated randomly and cannot be relied on between iterations of the product.

4.2.2.5 UUID_ATTRIBUTE_WITH_APPROXIMATE_LOCATION

The `uuid_attribute_with_approximate_location` is a subtype of `uuid_attribute` that provides an approximate location in cartesian space of an item or items that has a UUID assigned.

4.2.2.6 UUID_RELATIONSHIP

The `uuid_relationship` relates two UUIDs and provides a role for that relationship.

4.2.2.7 UUID_RELATIONSHIP_ROLE

The `uuid_relationship_role` enumerates the permitted roles associated with a `uuid_relationship`. The allowed roles are SUPERSEDES, MERGE, SPLIT, DERIVE_FROM, SAME_AS, and SIMILAR_TO.

4.2.2.8 UUID_PROVENANCE

The `uuid_provenance` is the specification of a sequence of `uuid_relationships` that provides a historical record of those relationships. The sequence is a simple list form.

4.2.2.9 UUID_CONTEXT_ROLE

The `uuid_context_role` associates a role (non-empty string value) to a UUID. Preprocessors are recommended to populate this string value as “design_iteration” or “downstream_manufacturing”, as appropriate.

4.2.2.10 uuid_tree_node

A uuid_tree_node is one of (uuid_leaf_node, uuid_internal_node, uuid_root_node). There are two optional attributes node_1 and node_2 that specify lower level nodes. A uuid_tree_node is ABSTRACT and shall not be populated by itself.

4.2.2.11 uuid_leaf_node

A uuid_leaf_node is a subtype of uuid_tree_node that is a leaf node in a tree. A uuid_leaf_node specifies a uuid_attribute as its data attribute. A uuid_leaf_node shall be referenced by two uuid_tree_nodes (but not itself or another uuid_leaf_node).

4.2.2.12 uuid_internal_node

A uuid_internal_node is a subtype of uuid_tree_node that is internal to the tree. It may be referenced by one uuid node (that is not a uuid_leaf_node) and shall reference two lower level nodes.

Note: For initial trial the merkle tree is out of scope. No tree entities need be populated.

4.2.2.13 uuid_root_node

A uuid_root_node is a subtype of uuid_tree_node that is the root node in a tree. It shall not be referenced by other nodes and shall reference two lower level nodes.The uuid_root_node provides the hash function for the tree.

4.2.3 UUID PREPROCESSOR AND POSTPROCESSOR RECOMMENDATIONS

Preprocessor Recommendations: All preprocessors must generate UUIDs for each entity that they wish to permanently identify.

Each Product entity ***must*** have a UUID assigned. Each Semantic PMI entity -- dimensions, tolerances, datum tags and targets, surface finishes, and model notes -- may have a UUID assigned. All topological entities such as faces, edges, and vertices as well as supplemental geometry entities that are used as reference for the above Semantic PMI entities in the pre-processing system may have UUIDs assigned. In addition, User Defined Attributes (UDAs) may be assigned to product, geometry, or PMI in support of the metrology use case and these UDAs may also have UUIDs assigned.

Preprocessors must ensure that all UUIDs assigned to entities from the CAD model as described above must be maintained and be stable from one iteration of the CAD model to the next, i.e. an entity will retain the same UUID from model iteration to iteration, from CAD session to session, or from a session on machine X to a session on a different machine Y as long as the entity exists in the data. This rule applies at all levels, i.e., to Product as well as to annotation (PMI) entities, and to associated geometry and topology or supplemental geometry entities. When new entities are created by the CAD user, new UUIDs will be assigned to those new entities. When entities are deleted from the model, their UUIDs ***must not be reused***.

It is recommended that pre-processors that own the original CAD data will publish UUIDs as described in Section 4.1 above as identifiers within the Data Section of the STEP file using the structures described in Section 4.2.2 above. Pre-processors may, however, use the Anchor Section method as an alternate approach.

Postprocessor Recommendations: Postprocessors must support the reading of UUIDs whether those UUIDs are published within the Data Section or within the Anchor Section. Postprocessors must retain incoming UUIDs for all identified geometry and topological entities, supplemental geometry entities, PMI entities, UDA entities and product entities read.

Related Entities:

N/A

4.2.4 PREPROCESSOR AND POSTPROCESSOR RECOMMENDATIONS FOR DESIGN ITERATION

Preprocessor Recommendations:

All preprocessors must generate UUIDs for each entity that they wish to permanently identify and must ensure that all UUIDs assigned to entities from the CAD model as described above must be maintained and be stable from one iteration of the CAD model to the next. (Ref – Section 4.2.2.11 above).

All preprocessors ***must export versioning information*** as specified Section 4.2.1 above.

Postprocessor Recommendations:

Initial Import -

All postprocessors, upon first import of a STEP model, must retain incoming UUIDs for all identified STEP entities and map UUIDs assigned to the equivalent internal CAD entities as imported. How this mapping is handled in each postprocessing application is left to the individual application to manage but a table of mappings should be retained for later use in subsequent iteration.

All postprocessors, upon the first import of the STEP model, must retain in the mapping the version information specified in Section 4.2.1 above as well as the product UUID and STEP file name information for the model for later use in subsequent iteration.

Subsequent to postprocessing, the imported model can be used for further design including the addition of new entities (geometry, features, datums, PMI, UDAs, process information, etc) ***as long as none of the imported entities that had UUIDs assigned are modified in any way.***

The postprocessing system shall, if new content is added, create and assign UUIDs to new entities as needed to further facilitate iterative exchange using the method described above for preprocessors.

Subsequent Import -

Subsequent to the initial import above, a postprocessor will, when loading a new STEP file for which an existing imported model exists in memory (i.e., that can be identified as having matching product UUIDs and file name information), replace all existing imported entities with the newly imported content, but will ensure that any native entities previously added will have their references preserved (reassigned to the same imported entities). This process will allow automated update of child references, thus preserving design intent from iteration to iteration.

Related Entities:

N/A

5 Express Diagrams

The EXPRESS entities and attributes used to support the complete requirements of entity identification for product, PMI, topology/geometry, supplemental geometry, and UDA are illustrated in the figures on the following pages. Note that all instances of terms in the diagrams shown in Figures 2 through 8 below having the characters “guid” in them are now replaced with the characters “uuid”.

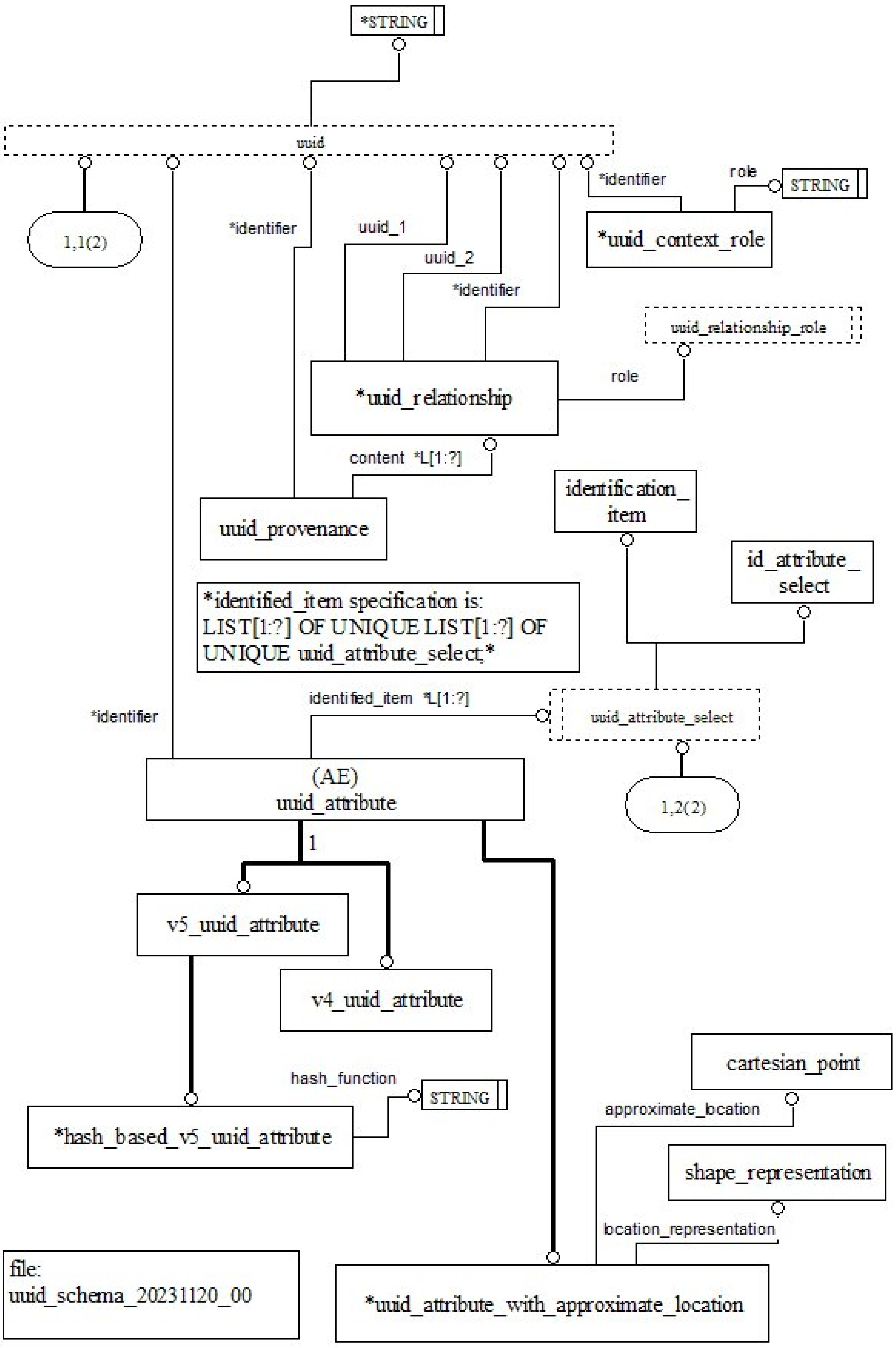

Figure 2: UUID Attribute Specific Schema Elements

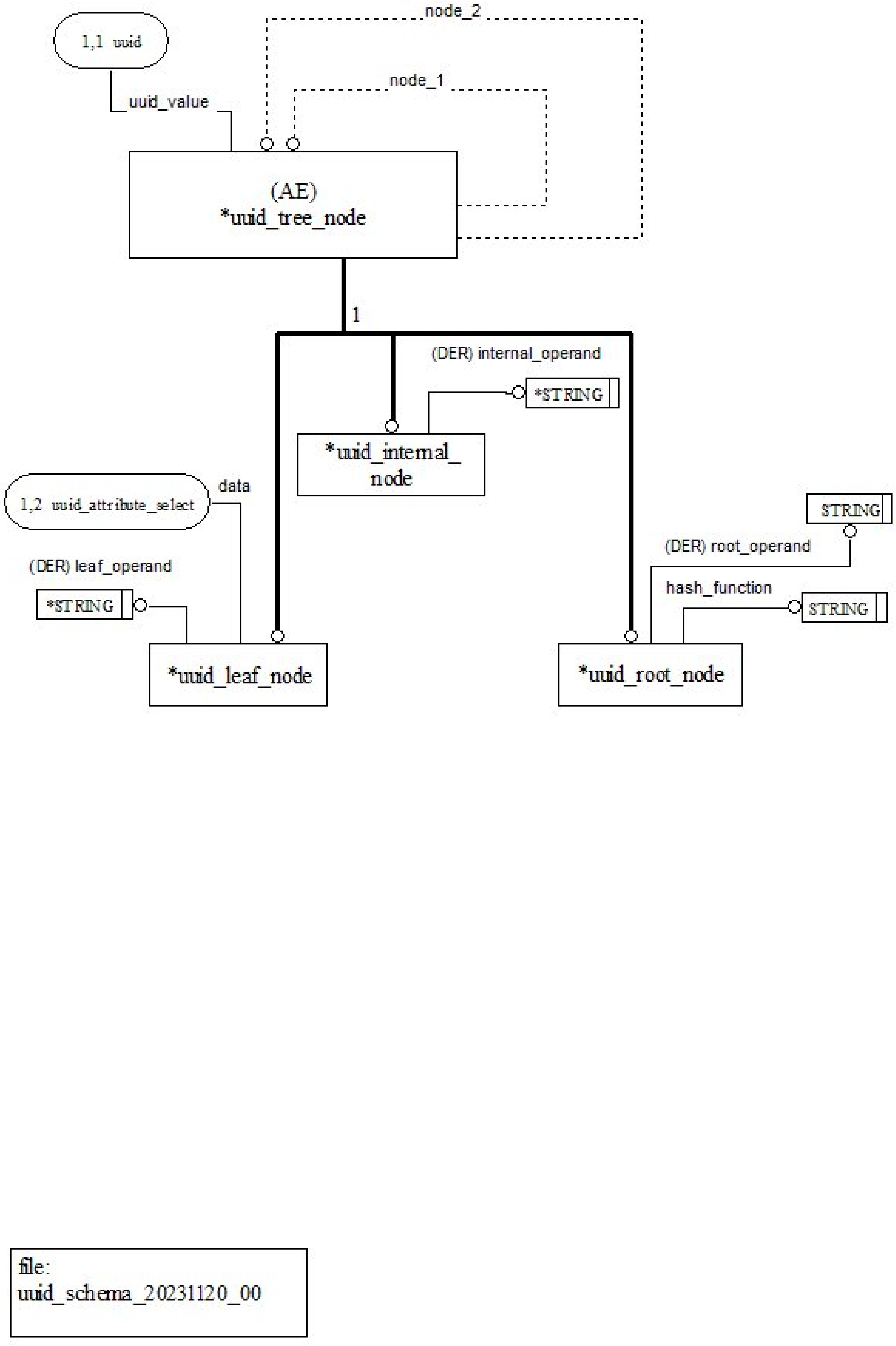

Figure 3: UUID Tree Specific Schema Elements

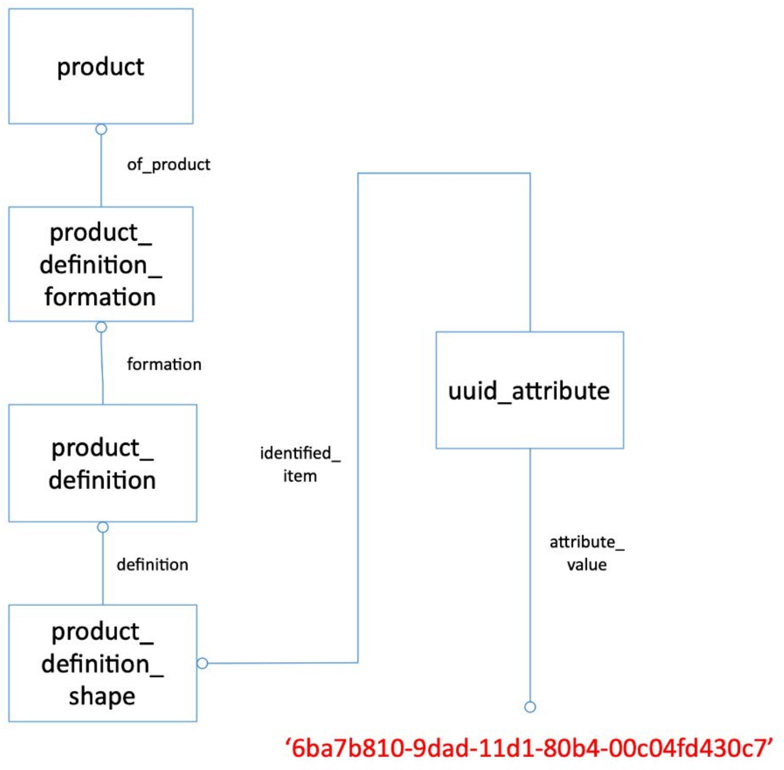

Figure 4: Entity Identifier for Product

(note – uuid_attribute must be replaced by either v5_uuid_attribute or v4_uuid_attribute.)

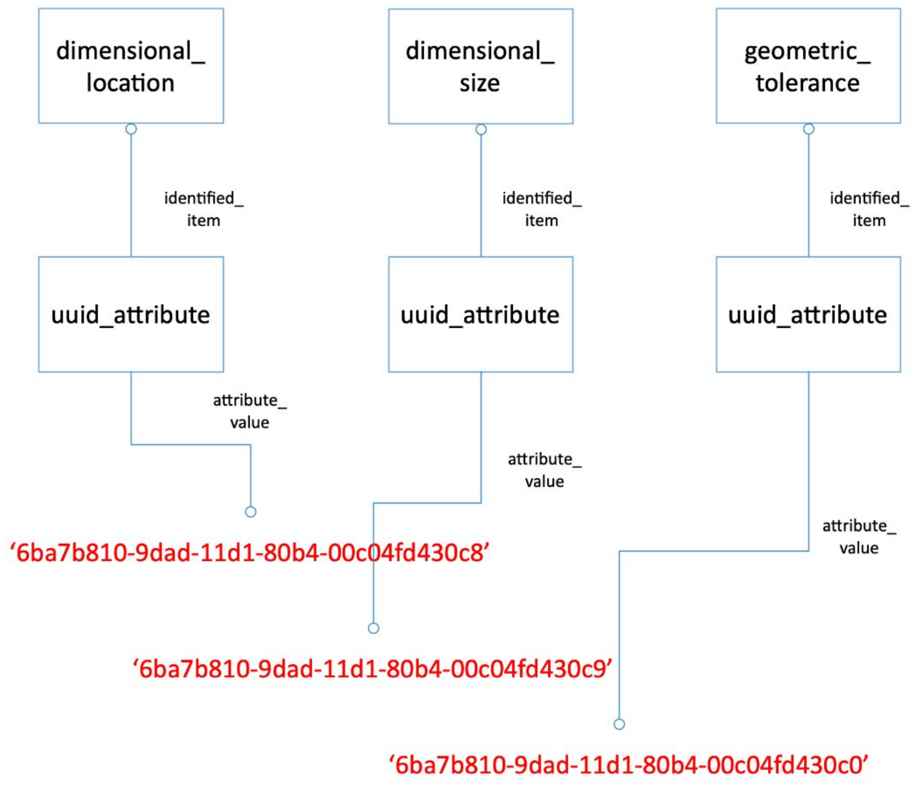

Figure 5: Entity Identifiers for PMI

(examples) (note – uuid_attribute must be replaced by either v5_uuid_attribute or v4_uuid_attribute.):

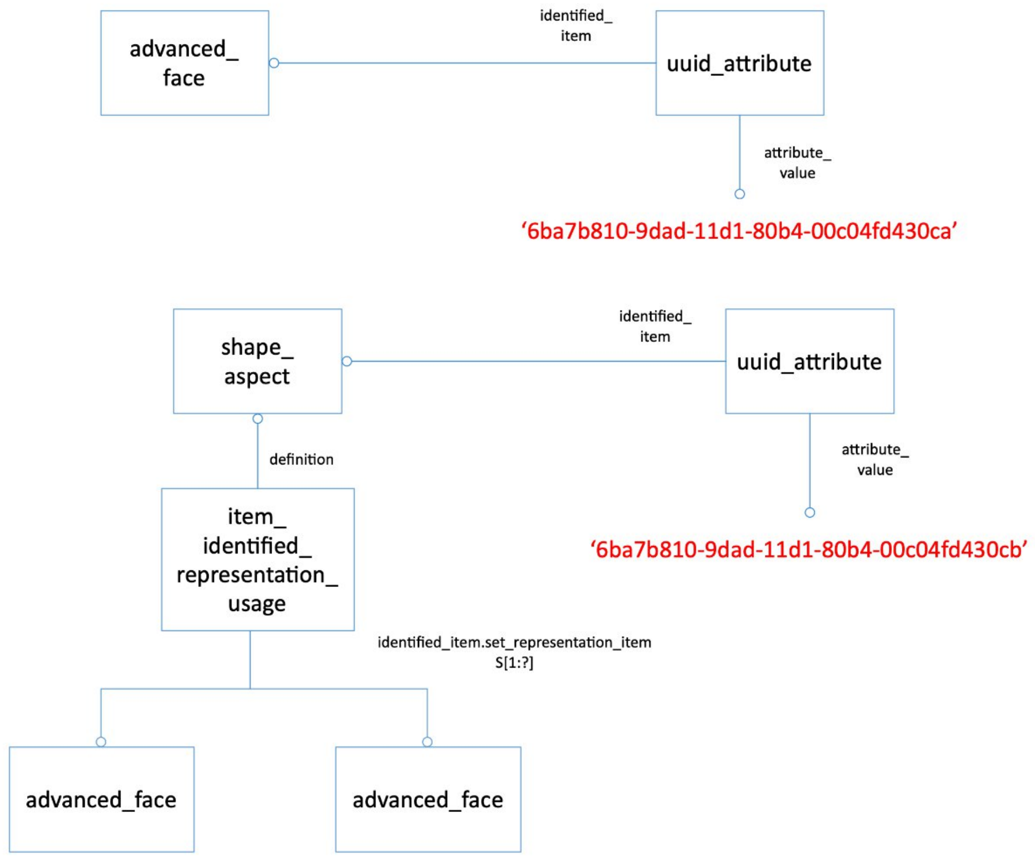
advanced_ face
identified_ item
uuid_attribute
attribute_ value
'6ba7b810-9dad-11d1-80b4-00c04fd430ca'
shape_ aspect
identified_ item
uuid_attribute
attribute_ value
'6ba7b810-9dad-11d1-80b4-00c04fd430cb'
definition
item_ identified_ representation_ usage
identified_item.set_representation_item S[1:?]
advanced_face
advanced_face

Figure 6: Entity Identifier for Topology/Geometry

(examples, including shape aspect for aggregation) (note – uuid_attribute must be replaced by either v5_uuid_attribute or v4_uuid_attribute.)

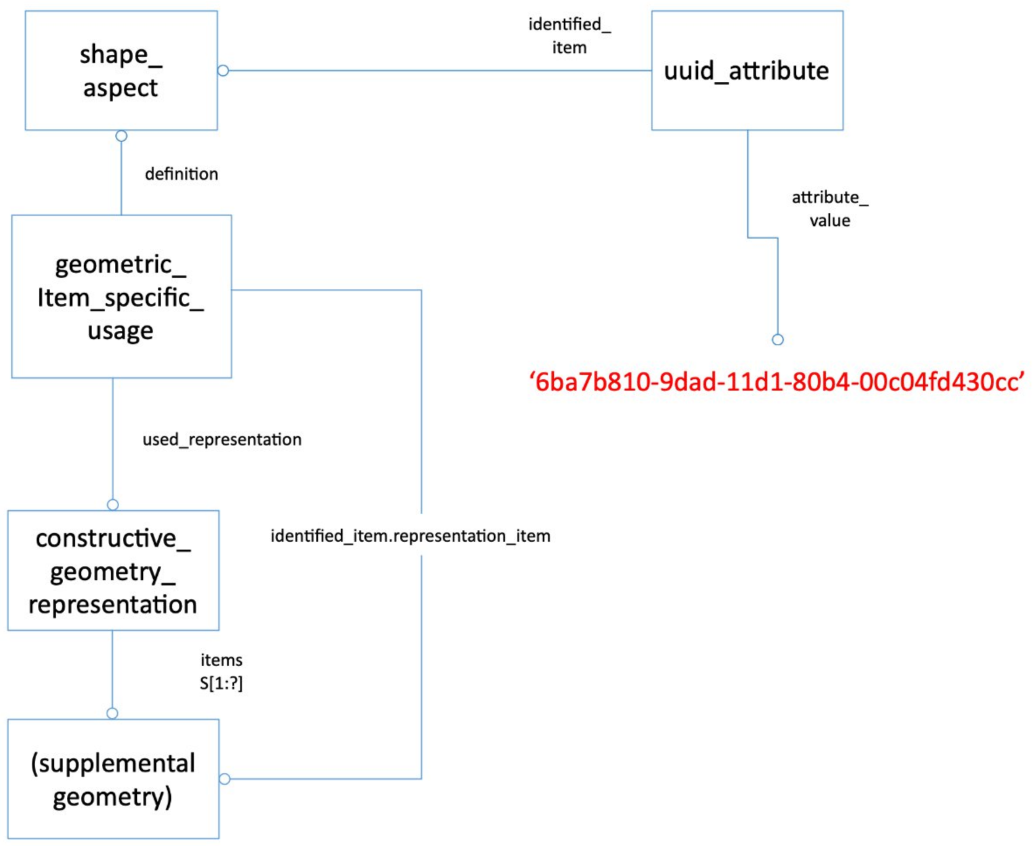
shape_ aspect
identified_ item
uuid_attribute
definition
attribute_ value
geometric_ Item_specific_ usage
'6ba7b810-9dad-11d1-80b4-00c04fd430cc'
used_representation
identified_item.representation_item
constructive_ geometry_ representation
items S[1:?]
(supplemental geometry)

Figure 7: Entity Identifier for Supplemental Geometry

(note – uuid_attribute must be replaced by either v5_uuid_attribute or v4_uuid_attribute.)

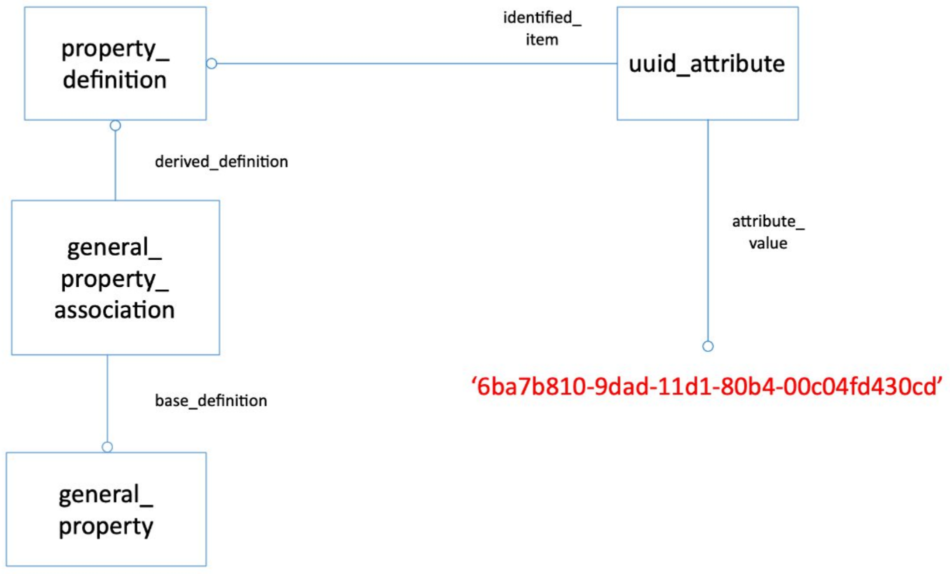

Figure 8: Entity Identifier for UDA

(note – uuid_attribute must be replaced by either v5_uuid_attribute or v4_uuid_attribute.)

Availability of Implementation Schemas

A.1 AP242 Edition 1

The long form EXPRESS schema for the first edition of AP242 can be retrieved from:

- http://www.cax-if.de/documents/ap242_is_mim_lf_v1.36.zip

A.2 AP242 Edition 2

The long form EXPRESS schema for the second edition of AP242 can be retrieved from:

- https://www.cax-if.de/documents/ap242ed2_mim_lf_v1.101.exp

A.3 AP242 Edition 3

The long form EXPRESS schema for the third edition of AP242 can be retrieved from:

- *https://www.cax-if.de/documents/ap242ed3_mim_lf_v1.152.exp*

A.4 AP 242 trial schema

The long form EXPRESS schema (based on AP 242 Edition 3) to be used for this recommended practice can be retrieved from

https://github.com/allisonfeeney/guid-data/blob/main/442_mim_lf_schema/mim_lf.exp