PRELIMINARY WORK ITEM International Standard

ISO/PWI 10303-81

2024-10-10

ISO/TC **184**/SC **4**/WG **12**

Secretariat: **ANSI**

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

Part 81: Description methods: The EXPRESS-Q language reference manual

Systèmes d'automatisation industrielle et intégration — Représentation et échange de données de produits —

Partie 81: Méthodes de description: Manuel de référence du langage EXPRESS-Q

Reference number
ISO/PWI 10303-81(en)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

NOTE This document is developed and published by the EXPRESS Language Foundation. All rights reserved.

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC 4, *Industrial data*.

ISO 10303 is organized as a series of parts, each published separately. The structure of ISO 10303 is described in ISO 10303–1.

Each part of ISO 10303 is a member of one of the following series: description methods, implementation methods, conformance testing methodology and framework, integrated generic resources, integrated application resources, application protocols, abstract test suites, application interpreted constructs, and application modules. This part is a member of the description methods series.

Introduction

EXPRESS-Q is a query language designed to work with EXPRESS schemas, particularly in the context of STEP (Standard for the Exchange of Product model data) application protocols. It provides a formal mechanism for defining mappings between Application Reference Model (ARM) and Model Implementation Model (MIM) schemas, as well as specifying constraints on datasets that comply with these schemas.

The primary purpose of EXPRESS-Q is to bridge the gap between conceptual models (ARM) and their implementations (MIM), ensuring that data exchange between different systems maintains semantic consistency. By providing a standardized way to express these mappings and constraints, EXPRESS-Q facilitates more robust and reliable data exchange in complex engineering and manufacturing environments.

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

Part 81:
Description methods: The EXPRESS-Q language reference manual

1 Scope

This document specifies EXPRESS-Q, including its syntax, semantics, and usage within the context of EXPRESS schemas and STEP application protocols.

The specification covers:

a) The grammar and syntax of EXPRESS-Q

b) The semantics of EXPRESS-Q constructs

c) The reference path syntax used for defining detailed mappings

d) Validation rules for EXPRESS-Q files

e) Guidelines for implementing EXPRESS-Q parsers and validators

This specification is intended for use by software developers, data modelers, and systems integrators working with EXPRESS schemas and STEP application protocols.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 10303-1:2024, *Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles*

ISO 10303-11:2004, *Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual*

ISO/IEC 8824-1:2008[1)], *Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation — Part 1:*

ISO/IEC 10646:2011[2)], *Information technology — Universal Coded Character Set (UCS)*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

1) Cancelled and replaced by ISO/IEC 8824-1:2015.

2) Cancelled and replaced by ISO/IEC 10646:2012.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org

3.1
EXPRESS-Q
query language for EXPRESS schemas, used to define mappings and constraints between ARM and MIM schemas

4 EXPRESS-Q language specification

4.1 Overview

EXPRESS-Q is designed to work alongside EXPRESS schemas, providing a mechanism to define mappings between ARM and MIM schemas, as well as specifying constraints on datasets. The language uses a syntax similar to EXPRESS but introduces new constructs specific to querying and mapping.

An EXPRESS-Q file typically consists of one or more entity mappings, each of which defines how an ARM entity and its attributes correspond to MIM elements. These mappings can include various declarations such as extensibility, AIM elements, sources, EXPRESS references, reference paths, and alternative mappings.

4.2 EBNF grammar

The following EBNF (Extended Backus-Naur Form) grammar defines the syntax of EXPRESS-Q:

```
express_q_file = { entity_mapping } ;

entity_mapping = "ENTITY_MAPPING" entity_name ";" newline
                 [ extensible_decl ]
                 [ aim_element_decl ]
                 [ source_decl ]
                 [ express_ref_decl ]
                 [ refpath_decl ]
                 [ alt_map_decl ]
                 { attribute_mapping }
                 "END_ENTITY_MAPPING" ";" ;

extensible_decl = "EXTENSIBLE" ":" boolean ";" newline ;

aim_element_decl = "AIM_ELEMENT" ":" string ";" newline ;

source_decl = "SOURCE" ":" string ";" newline ;

express_ref_decl = "EXPRESS_REF" ":" "[" [ string { "," string } ] "]" ";" newline ;

refpath_decl = "REFPATH" ":" "{" newline
               refpath_content
               "}" ";" newline ;

refpath_content = { refpath_line } ;

refpath_line = [ indentation ] reference_expression newline ;

reference_expression = entity_reference [ relationship { relationship } ] ;

relationship = path_relation | supertype_relation | subtype_relation |
                select_extension | constraint ;

entity_reference = identifier ;

path_relation = "->" entity_reference ;
```

```
supertype_relation = "=>" entity_reference ;

subtype_relation = "<=" entity_reference ;

select_extension = "*" ">" entity_reference | "<" "*" entity_reference ;

constraint = "[" constraint_expression "]" ;

constraint_expression = ? expression following EXPRESS syntax ? ;

indentation = ? one or more spaces or tabs ? ;

alt_map_decl = "ALT_MAP" ":" "[" [ string { "," string } ] "]" ";" newline ;

attribute_mapping = "ATTRIBUTE_MAPPING" attribute_name ";" newline
                    "ASSERTION_TO" ":" string ";" newline
                    [ aim_element_decl ]
                    [ source_decl ]
                    [ express_ref_decl ]
                    [ refpath_decl ]
                    "END_ATTRIBUTE_MAPPING" ";" newline ;

entity_name = identifier ;

attribute_name = identifier ;

identifier = letter { letter | digit | "_" } ;

boolean = "TRUE" | "FALSE" ;

string = '"' { character } '"' ;

letter = "A" | ... | "Z" | "a" | ... | "z" ;

digit = "0" | ... | "9" ;

character = ? any printable character ? ;

indentation = ? one or more spaces or tabs ? ;

newline = ? line break character(s) ? ;
```

Figure 1

This grammar defines the overall structure of an EXPRESS-Q file, including entity mappings, attribute mappings, reference paths, including various types of relationships between entities and constraints, and various declarations.

4.3 Semantics

This section describes the meaning and usage of each major construct in EXPRESS-Q.

4.3.1 Entity mapping

```
entity_mapping = "ENTITY_MAPPING" entity_name ";" newline
                 [ extensible_decl ]
                 [ aim_element_decl ]
                 [ source_decl ]
                 [ express_ref_decl ]
                 [ refpath_decl ]
                 [ alt_map_decl ]
                 { attribute_mapping }
```

```
"END_ENTITY_MAPPING" ";" ;
```

Figure 2

An entity mapping defines how an ARM entity corresponds to one or more MIM entities or constructs. It encapsulates all the information needed to map a single ARM entity to its MIM counterpart(s).

Usage

— Begin with “ENTITY_MAPPING” followed by the ARM entity name.

— Include optional declarations for extensibility, AIM element, source, EXPRESS references, reference path, and alternative mappings.

— Include attribute mappings for the entity’s attributes.

— End with “END_ENTITY_MAPPING”.

```
ENTITY_MAPPING Product;
EXTENSIBLE: FALSE;
AIM_ELEMENT: "product";
SOURCE: "ISO 10303-41";
EXPRESS_REF: ["product_definition_schema"];
REFPATH: {
  product <= product_definition_formation
  product_definition_formation.of_product -> product
};
ATTRIBUTE_MAPPING id;
ASSERTION_TO: "id";
AIM_ELEMENT: "product.id";
END_ATTRIBUTE_MAPPING;
END_ENTITY_MAPPING;
```

Figure 3

4.3.2 Extensible declaration

```
extensible_decl = "EXTENSIBLE" ":" boolean ";" newline ;
```

Figure 4

The extensible declaration specifies whether the entity mapping is extensible, corresponding to the extensibility of SELECT types in EXPRESS schemas.

Usage

— Use “EXTENSIBLE: TRUE;” for extensible mappings.

— Use “EXTENSIBLE: FALSE;” for non-extensible mappings.

```
ENTITY_MAPPING ProductCategory;
EXTENSIBLE: TRUE;
...
END_ENTITY_MAPPING;
```

Figure 5

4.3.3 AIM element declaration

```
aim_element_decl = "AIM_ELEMENT" ":" string ";" newline ;
```

Figure 6

The AIM element declaration specifies the corresponding AIM (MIM) element for the ARM entity, providing a direct link between the ARM and MIM schemas.

Usage

— Specify the AIM element name as a string.

```
ENTITY_MAPPING Product;
AIM_ELEMENT: "product";
...
END_ENTITY_MAPPING;
```

Figure 7

4.3.4 Source declaration

```
source_decl = "SOURCE" ":" string ";" newline ;
```

Figure 8

The source declaration indicates the source (typically an ISO standard) of the MIM element, which is crucial for traceability and maintaining consistency with relevant standards.

Usage

— Specify the source as a string, typically an ISO standard number.

```
ENTITY_MAPPING Product;
SOURCE: "ISO 10303-41";
...
END_ENTITY_MAPPING;
```

Figure 9

4.3.5 EXPRESS reference declaration

```
express_ref_decl = "EXPRESS_REF" ":" "[" [ string { "," string } ] "]" ";" newline ;
```

Figure 10

The EXPRESS reference declaration lists EXPRESS references related to the mapping, used to link the mapping to specific parts of the EXPRESS schemas or other relevant documentation.

Usage

— Provide a comma-separated list of references enclosed in square brackets.

— Use an empty list “[]” if there are no references.

```
ENTITY_MAPPING Product;
EXPRESS_REF: ["product_definition_schema", "management_resources_schema"];
...
END_ENTITY_MAPPING;
```

Figure 11

4.3.6 Reference path declaration

4.3.6.1 General

The reference path syntax in EXPRESS-Q is based on the ISO 10303 Mapping Specification and is used within the REFPATH sections to define precise mappings between ARM and MIM elements. This syntax allows for complex relationships and constraints to be expressed concisely.

```
refpath_decl = "REFPATH" ":" "{" newline
               refpath_content
```

```
"}" ";" newline ;
```

Figure 12

The reference path declaration defines the detailed mapping between ARM and MIM elements using the reference path syntax described earlier.

Usage

— Begin with “REFPATH: {” and end with “};”.

— Each line in the content describes a step in the mapping path.

— Use indentation to indicate nested structures.

— Use symbols like “⇐”, “⇒”, “→” to indicate relationships between entities and attributes.

```
ENTITY_MAPPING Product;
REFPATH: {
  product <= product_definition_formation
  product_definition_formation.of_product -> product
  {product.name -> product.name
   product.description -> product.description}
};
...
END_ENTITY_MAPPING;
```

Figure 13

4.3.6.2 Symbols and meaning

The reference path syntax uses several symbols to express relationships and constraints. These symbols are derived from the ISO 10303 Mapping Specification:

`[]` Enclosed section constrains multiple MIM elements or sections of the reference path required to satisfy an information requirement.

`()` Enclosed section constrains multiple MIM elements or sections of the reference path identified as alternatives within the mapping.

`{}` Enclosed section constrains the reference path to satisfy an information requirement.

`<>` Enclosed section constrains one or more required reference paths.

`||` Enclosed section constrains the supertype entity.

→ The attribute whose name precedes the symbol references the entity or select type whose name follows the symbol.

← The entity or select type whose name precedes the symbol is referenced by the entity attribute whose name follows the symbol.

`[i]` The attribute whose name precedes the symbol is an aggregate; any element of that aggregate is referred to.

`[n]` The attribute whose name precedes the symbol is an ordered aggregate; member n of that aggregate is referred to.

⇒ The entity whose name precedes the symbol is a supertype of the entity whose name follows the symbol.

⇐ The entity whose name precedes the symbol is a subtype of the entity whose name follows the symbol.

= The string, select, or enumeration type is constrained to a choice or value.

\ The reference path expression continues on the next line.

* One or more instances of the relationship entity data type may be assembled in a relationship tree structure.

-- The text following is a comment or introduces a clause reference.

*> The select or enumeration type whose name precedes the symbol is extended into the select or enumeration type whose name follows the symbol.

<* The select or enumeration type whose name precedes the symbol is an extension of the select or enumeration type whose name follows the symbol.

! Section enclosed by {} indicates a negative constraint placed on the mapping.
{}

4.3.6.3 Usage and examples

The reference path syntax is used to define precise mappings between ARM and MIM elements. Here are some examples demonstrating various aspects of the syntax:

EXAMPLE — Simple attribute mapping

```
    REFPATH: {
      product.id -> id
    }
```

```
REFPATH: {
  mechanical_context <= product_context
}
```

Figure 14 — Subtype relationship

```
REFPATH: {
  shape_representation <= representation
  {representation.items[i] -> representation_item
   representation_item =>
   (geometric_representation_item
    [geometric_representation_item.dim = 3] |
    mapped_item)
  }
}
```

Figure 15 — Complex mapping with alternatives and constraints

```
REFPATH: {
  person_and_organization_select *> organization
}
```

Figure 16 — Select type extension

```
REFPATH: {
  product_definition_formation
  !{product_definition_formation.make_or_buy = bought_item}
}
```

Figure 17 — Negative constraint

4.3.6.4 Validation rules for reference paths

When validating reference paths, the following rules should be applied:

a) All referenced entities and attributes must exist in the corresponding ARM or MIM schema.

b) The relationships between entities (supertype, subtype, attribute reference) must be consistent with the schema definitions.

c) Constraints must be syntactically correct and use only attributes or functions that are valid for the constrained entity.

d) Select type extensions must be consistent with the schema definitions.

e) Aggregate references (using `[i]` or `[n]`) must be applied only to attributes defined as aggregates in the schema.

f) Alternative mappings (using parentheses) must all be valid according to the schema.

g) Negative constraints must be logically consistent with the rest of the mapping and the schema definitions.

h) The overall structure of the reference path must form a valid path through the schema, connecting the source ARM entity to the target MIM entity or attribute.

4.3.7 Alternative mapping declaration

```
alt_map_decl = "ALT_MAP" ":" "[" [ string { "," string } ] "]" ";" newline ;
```

Figure 18

The alternative mapping declaration specifies alternative mappings, allowing for flexibility in cases where multiple valid mappings exist.

Usage

— Provide a comma-separated list of alternative mappings enclosed in square brackets.

— Use an empty list "[]" if there are no alternative mappings.

```
ENTITY_MAPPING GeometricRepresentation;
ALT_MAP: ["shape_representation", "tessellated_shape_representation"];
...
END_ENTITY_MAPPING;
```

Figure 19

4.3.8 Attribute mapping

```
attribute_mapping = "ATTRIBUTE_MAPPING" attribute_name ";" newline
                    "ASSERTION_TO" ":" string ";" newline
                    [ aim_element_decl ]
                    [ source_decl ]
                    [ express_ref_decl ]
                    [ refpath_decl ]
                    "END_ATTRIBUTE_MAPPING" ";" newline ;
```

Figure 20

An attribute mapping defines how an individual attribute of an ARM entity corresponds to elements in the MIM schema.

Usage

— Begin with "ATTRIBUTE_MAPPING" followed by the attribute name.

— Specify the ASSERTION_TO value, which indicates the corresponding MIM element or type.

— Optionally include AIM_ELEMENT, SOURCE, EXPRESS_REF, and REFPATH declarations.

— End with "END_ATTRIBUTE_MAPPING".

```
ENTITY_MAPPING Product;
ATTRIBUTE_MAPPING id;
ASSERTION_TO: "id";
AIM_ELEMENT: "product.id";
REFPATH: {
  product.id -> id
};
END_ATTRIBUTE_MAPPING;

ATTRIBUTE_MAPPING name;
ASSERTION_TO: "label";
AIM_ELEMENT: "product.name";
SOURCE: "ISO 10303-41";
EXPRESS_REF: ["product_definition_schema"];
REFPATH: {
  product.name -> name
};
END_ATTRIBUTE_MAPPING;
END_ENTITY_MAPPING;
```

Figure 21

5 Validation rules

5.1 General

Validation of EXPRESS-Q files is crucial to ensure the correctness and consistency of mappings between ARM and MIM schemas. The following rules should be applied when validating EXPRESS-Q files.

5.2 Entity mapping validation

a) Each ENTITY_MAPPING must correspond to an entity defined in the ARM schema.

b) The AIM_ELEMENT specified must exist in the MIM schema.

c) The SOURCE referenced must be a valid ISO standard or other recognized source.

d) All EXPRESS_REF entries must refer to valid EXPRESS schemas.

e) The REFPATH must form a valid path from the ARM entity to the specified AIM_ELEMENT.

f) If ALT_MAP is specified, all alternative mappings must be valid MIM entities.

5.3 Attribute mapping validation

a) Each ATTRIBUTE_MAPPING must correspond to an attribute of the ARM entity being mapped.

b) The ASSERTION_TO value must be a valid type or entity in the MIM schema.

c) If specified, the AIM_ELEMENT must exist in the MIM schema.

d) The REFPATH for an attribute must form a valid path from the ARM attribute to the specified MIM element.

5.4 Reference path validation

a) All entities and attributes referenced in the REFPATH must exist in either the ARM or MIM schema, as appropriate.

b) Relationships between entities (e.g., subtype, supertype) must be consistent with the schema definitions.

c) Constraints specified in the REFPATH must use valid attributes or functions for the entities they constrain.

d) SELECT type extensions must be consistent with the schema definitions.

e) Aggregate references (using `[i]` or `[n]`) must only be applied to attributes defined as aggregates in the schema.

5.5 Overall consistency

a) The complete set of mappings must cover all entities and attributes in the ARM schema.

b) There should be no conflicting mappings (e.g., two different mappings for the same ARM entity or attribute).

c) The overall set of mappings should form a consistent and complete transformation from the ARM to the MIM schema.

6 Implementation guidelines

6.1 General

When implementing an EXPRESS-Q parser and validator, consider the following guidelines.

6.2 Parsing

a) Implement a lexical analyzer to tokenize the EXPRESS-Q input.

b) Develop a parser based on the EBNF grammar provided in this specification.

c) Create an abstract syntax tree (AST) or similar internal representation of the parsed EXPRESS-Q file.

6.3 Schema loading

a) Implement an EXPRESS schema parser or use an existing library to load and interpret ARM and MIM schemas.

b) Create internal representations of the schemas that allow for efficient querying of entities, attributes, and relationships.

6.4 Validation

a) Implement validation checks based on the rules outlined in the “Validation rules” section.

b) Perform two-pass validation:

   1) First pass: Check syntax and basic semantic rules (e.g., entity and attribute existence).

   2) Second pass: Validate complex semantic rules (e.g., consistency of reference paths).

c) Provide clear and informative error messages for validation failures.

6.5 Reference path interpretation

a) Implement a separate parser for the reference path syntax.

b) Develop algorithms to traverse reference paths within the context of the loaded schemas.

c) Implement functions to evaluate constraints specified in reference paths.

6.6 Performance considerations

a) Use efficient data structures (e.g., hash tables) for quick lookups of schema elements.

b) Implement caching mechanisms for frequently accessed schema information.

c) Consider parallel processing for validating multiple entity or attribute mappings simultaneously.

6.7 Extensibility

a) Design the implementation to be extensible, allowing for future enhancements to the EXPRESS-Q language.

b) Use modular design to separate concerns (e.g., parsing, schema handling, validation).

6.8 Integration

a) Provide APIs or interfaces for integrating the EXPRESS-Q parser and validator into larger systems or workflows.

b) Consider implementing export functionality to generate reports or machine-readable representations of the validation results.

7 Use cases and examples

7.1 General

This section provides examples of how EXPRESS-Q can be used in various scenarios related to STEP application protocols.

7.2 Use case 1: Mapping a complex product structure

In this example, we'll demonstrate how EXPRESS-Q can be used to map a complex product structure from an ARM schema to a MIM schema.

```
ENTITY_MAPPING Product;
AIM_ELEMENT: "product";
SOURCE: "ISO 10303-41";
REFPATH: {
  product <= product_definition_formation
  product_definition_formation.of_product -> product
};

ATTRIBUTE_MAPPING id;
ASSERTION_TO: "id";
AIM_ELEMENT: "product.id";
END_ATTRIBUTE_MAPPING;

ATTRIBUTE_MAPPING name;
ASSERTION_TO: "label";
AIM_ELEMENT: "product.name";
END_ATTRIBUTE_MAPPING;

ATTRIBUTE_MAPPING description;
ASSERTION_TO: "text";
AIM_ELEMENT: "product.description";
END_ATTRIBUTE_MAPPING;

ATTRIBUTE_MAPPING components;
ASSERTION_TO: "SET OF Component";
REFPATH: {
  product <- product_definition_formation
  product_definition_formation <- product_definition
  product_definition <- product_definition_relationship.relating_product_definition
  product_definition_relationship ->
  product_definition_relationship.related_product_definition
  product_definition_relationship.related_product_definition -> product_definition
  product_definition -> product_definition_formation
```

```
  product_definition_formation -> product
};
END_ATTRIBUTE_MAPPING;

END_ENTITY_MAPPING;

ENTITY_MAPPING Component;
AIM_ELEMENT: "product";
SOURCE: "ISO 10303-41";
REFPATH: {
  product <= product_definition_formation
  product_definition_formation.of_product -> product
};

ATTRIBUTE_MAPPING id;
ASSERTION_TO: "id";
AIM_ELEMENT: "product.id";
END_ATTRIBUTE_MAPPING;

ATTRIBUTE_MAPPING name;
ASSERTION_TO: "label";
AIM_ELEMENT: "product.name";
END_ATTRIBUTE_MAPPING;

ATTRIBUTE_MAPPING quantity;
ASSERTION_TO: "measure_with_unit";
REFPATH: {
  product <- product_definition_formation
  product_definition_formation <- product_definition
  product_definition <- product_definition_relationship
  product_definition_relationship.related_product_definition_quantity -> measure_with_unit
};
END_ATTRIBUTE_MAPPING;

END_ENTITY_MAPPING;
```

Figure 22

This example demonstrates how a product with multiple components can be mapped from an ARM schema to a MIM schema using EXPRESS-Q. It shows the use of complex reference paths to navigate relationships between entities in the MIM schema.

7.3 Use case 2: Mapping with alternative representations

This example shows how EXPRESS-Q can handle cases where an ARM entity might be mapped to different MIM representations based on certain conditions.

```
ENTITY_MAPPING GeometricRepresentation;
ALT_MAP: ["shape_representation", "tessellated_shape_representation"];
SOURCE: "ISO 10303-42";

REFPATH: {
  (shape_representation <= representation
   [representation.context_of_items->geometric_representation_context]) |
  (tessellated_shape_representation <= representation
   [representation.context_of_items->geometric_representation_context])
};

ATTRIBUTE_MAPPING items;
ASSERTION_TO: "SET OF geometric_representation_item";
REFPATH: {
  representation.items[i] -> geometric_representation_item
};
END_ATTRIBUTE_MAPPING;
```

```
END_ENTITY_MAPPING;
```

Figure 23

This example demonstrates how EXPRESS-Q can handle alternative mappings for a single ARM entity. The GeometricRepresentation entity can be mapped to either a shape_representation or a tessellated_shape_representation in the MIM schema, depending on the specific requirements of the application.

8 Summary

EXPRESS-Q provides a powerful and flexible language for defining mappings between ARM and MIM schemas in the context of STEP application protocols. By offering a standardized way to express these mappings, EXPRESS-Q facilitates more robust and reliable data exchange in complex engineering and manufacturing environments.

Key features of EXPRESS-Q include:

a) A syntax that is familiar to users of EXPRESS, reducing the learning curve.

b) The ability to define detailed mappings between ARM and MIM entities and attributes.

c) Support for complex reference paths, allowing for navigation through intricate schema relationships.

d) Mechanisms for specifying constraints and alternative mappings.

e) Clear separation of entity and attribute mappings for improved readability and maintainability.

The validation rules and implementation guidelines provided in this specification aim to ensure consistent interpretation and implementation of EXPRESS-Q across different tools and systems.

9 Future considerations

As the use of EXPRESS-Q grows and evolves, several areas may be considered for future enhancements:

9.1 Extended constraint language

Future versions of EXPRESS-Q might benefit from a more expressive constraint language, allowing for more complex conditions and transformations to be specified within mappings.

9.2 Integration with other standards

Consideration should be given to how EXPRESS-Q can be integrated with other relevant standards and technologies, such as ontology languages or model transformation frameworks.

9.3 Tool support

Development of standardized libraries and tools for parsing, validating, and processing EXPRESS-Q files would greatly facilitate its adoption and use.

9.4 Reverse mapping capabilities

Future versions might include capabilities for defining bi-directional mappings, allowing for the generation of ARM instances from MIM data.

9.5 Performance optimizations

As EXPRESS-Q is used with increasingly large and complex schemas, performance optimizations in both the language design and implementation guidelines may become necessary.

Annex A
(informative)

Complete EXPRESS-Q file example

This annex provides a complete example of an EXPRESS-Q file, demonstrating various features of the language in a cohesive context.

```
-- Example EXPRESS-Q file for a simplified product data model

ENTITY_MAPPING Product;
EXTENSIBLE: FALSE;
AIM_ELEMENT: "product";
SOURCE: "ISO 10303-41";
EXPRESS_REF: ["product_definition_schema"];
REFPATH: {
  product <= product_definition_formation
  product_definition_formation.of_product -> product
};

ATTRIBUTE_MAPPING id;
ASSERTION_TO: "identifier";
AIM_ELEMENT: "product.id";
REFPATH: {
  product.id -> id
};
END_ATTRIBUTE_MAPPING;

ATTRIBUTE_MAPPING name;
ASSERTION_TO: "label";
AIM_ELEMENT: "product.name";
REFPATH: {
  product.name -> name
};
END_ATTRIBUTE_MAPPING;

ATTRIBUTE_MAPPING description;
ASSERTION_TO: "text";
AIM_ELEMENT: "product.description";
REFPATH: {
  product.description -> description
};
END_ATTRIBUTE_MAPPING;

ATTRIBUTE_MAPPING category;
ASSERTION_TO: "product_category";
REFPATH: {
  product <- product_category_assignment.products[i]
  product_category_assignment -> product_category_assignment.assigned_category
  product_category_assignment.assigned_category -> product_category
};
END_ATTRIBUTE_MAPPING;

END_ENTITY_MAPPING;

ENTITY_MAPPING Assembly;
EXTENSIBLE: FALSE;
AIM_ELEMENT: "product";
SOURCE: "ISO 10303-44";
EXPRESS_REF: ["product_structure_schema"];
```

```
REFPATH: {
  product <= product_definition_formation
  product_definition_formation.of_product -> product
  [product <- product_definition_formation
   product_definition_formation <- product_definition
   product_definition <- assembly_component_usage.relating_product_definition]
};

ATTRIBUTE_MAPPING components;
ASSERTION_TO: "SET OF AssemblyComponent";
REFPATH: {
  product <- product_definition_formation
  product_definition_formation <- product_definition
  product_definition <- assembly_component_usage.relating_product_definition
  assembly_component_usage -> assembly_component_usage.related_product_definition
  assembly_component_usage.related_product_definition -> product_definition
  product_definition -> product_definition_formation
  product_definition_formation -> product
};
END_ATTRIBUTE_MAPPING;

END_ENTITY_MAPPING;

ENTITY_MAPPING AssemblyComponent;
EXTENSIBLE: FALSE;
AIM_ELEMENT: "product";
SOURCE: "ISO 10303-44";
EXPRESS_REF: ["product_structure_schema"];
REFPATH: {
  product <= product_definition_formation
  product_definition_formation.of_product -> product
};

ATTRIBUTE_MAPPING quantity;
ASSERTION_TO: "measure_with_unit";
REFPATH: {
  product <- product_definition_formation
  product_definition_formation <- product_definition
  product_definition <- assembly_component_usage
  assembly_component_usage.quantity -> measure_with_unit
};
END_ATTRIBUTE_MAPPING;

END_ENTITY_MAPPING;

ENTITY_MAPPING GeometricRepresentation;
EXTENSIBLE: TRUE;
ALT_MAP: ["shape_representation", "tessellated_shape_representation"];
SOURCE: "ISO 10303-42";
EXPRESS_REF: ["geometric_model_schema"];

REFPATH: {
  (shape_representation <= representation
   [representation.context_of_items->geometric_representation_context]) |
  (tessellated_shape_representation <= representation
   [representation.context_of_items->geometric_representation_context])
};

ATTRIBUTE_MAPPING items;
ASSERTION_TO: "SET OF geometric_representation_item";
REFPATH: {
  representation.items[i] -> geometric_representation_item
};
END_ATTRIBUTE_MAPPING;
```

```
END_ENTITY_MAPPING;
```

Figure A.1

This example demonstrates a simplified product data model, including mappings for products, assemblies, assembly components, and geometric representations. It showcases various features of EXPRESS-Q, including:

— Entity and attribute mappings

— Reference paths with complex relationships

— Alternative mappings

— Constraints within reference paths

— Use of SELECT types (implied by the EXTENSIBLE: TRUE; in GeometricRepresentation)

This example is not exhaustive but provides a realistic scenario that illustrates how EXPRESS-Q can be used to define mappings between ARM and MIM schemas in a STEP application protocol context.

Bibliography

[1] ISO 10303-21:2016, *Industrial automation systems and integration — Product data representation and exchange — Part 21: Implementation methods: Clear text encoding of the exchange structure*

Price based on 17 pages