CLRC

Technical Report
RAL-TR-96-087

The MIDAS Data Model for Electromagnetic and Stress Analysis Integration

D Thomas and C Greenough

October 1996

COUNCIL FOR THE CENTRAL LABORATORY OF THE RESEARCH COUNCILS

Enquiries about copyright, reproduction and requests for
additional copies of this report should be addressed to:

The Central Laboratory of the Research Councils
Library and Information Services
Rutherford Appleton Laboratory
Chilton
Didcot
Oxfordshire
OX11 0QX
Tel: 01235 445384 Fax: 01235 446403
E-mail library@rl.ac.uk

ISSN 1358-6254

The MIDAS Data Model for Electromagnetic and Stress Analysis Integration

Version 1.2

Mrs D Thomas and Dr C Greenough
Rutherford Appleton Laboratory

October 1996

Abstract

The International Standard for data exchange and sharing ISO 10303, known as STEP, is now being used as an integration tool between CAD applications. The initial release of the Standard did not include Finite Element Analysis but more stable versions of the FEA Standard are available. Electromagnetic analysis is not covered presently by STEP but the existing models and methodology can be used and extended into this area.

MIDAS (ESPRIT project 7294) was an EC funded project which brought together software from different vendors in the areas of solid modelling, advanced mesh generation, stress analysis and electromagnetic analysis to provide an integrated suite of software with a common user interface and a common database. This integration used the STEP nethodology.

This report contains the data model used in integrating stress analysis and electromagnetic analysis together with the necessary simplified geometry. It is designed to be directly implementable into a simple database. The scope of the data model is defined. The changes that needed to be made to the STEP data models in order to be implemented into a database are described.

Mathematical Software Group
Department for Computation and Information
Rutherford Appleton Laboratory
Chilton, Didcot
Oxfordshire OX11 0QX

Contents

1 Introduction

This report presents a Data Model which has been developed for the purposes of integrating stress analysis and electromagnetic analysis packages via a common database together with the necessary simplified geometry obtained from a Computer Aided Design (CAD) system. The data model has been developed in the *EXPRESS* Information Modeling Language and is based on relevant STEP data models.

The first section details the scope of each part of the data model. The second section describes the changes that were necessary throughout the data model to relevant STEP entities to enable the resulting data model to be directly implementable into DEVA or any simple database.

The following sections contain the MIDAS data model itself. Together with sufficient explanation of the Entities and Attributes to enable implementation.

Appendix A lists those entities from STEP data models which have been used in the MIDAS model. Together with a indication of whether each entity has been used unchanged, changed slightly or changed significantly.

The related report [1] gives more details of the STEP standard, the MIDAS project, the way the data modelling needs of electromagnetic analysis were determined and the way that the STEP models were adapted and extended for use in MIDAS.

Version 1.1 of the MIDAS data model was presented in the MIDAS report, MIDAS.RAL.94.5 [2]. The version contained in this report, Version 1.2 includes the following improvements:

- addition of voids to boundary representation solids
- addition of entities to deal with requests for analysis output
- addition of entities to deal with analysis results
- minor corrections to entities for finite element mesh representation

2 Scope

This data model specifies the application entities for integrating geometry, stress analysis and electromagnetic analysis including simplified geometry, finite element models and analysis control information. It is partitioned into four areas.

2.1 Geometry

There are two main areas of MIDAS in which geometry is used. The full geometrical descriptions of the component are generated using the geometrical modeller. These are accessed by the analysis programs via the geometric modeller and there is no urgent need to include this data in the MIDAS database. This area will be considered later in the project. The second area of geometry is the simplified geometry which is used for input to the analysis packages. This consists only of a simple boundary representation.

The following are within scope in this schema:

- definition of points;
- definition of conic curves and elementary surfaces;

- definition of the fundamental topological entities vertex, edge and face, each with a specialised subtype to enable it to be associated with the geometry of a point, curve or surface, respectively;
- collections of the basic entities to form topological structures of path, loop and shell and constraints to ensure the integrity of these structures;
- orientation of topological entities;
- manifold boundary representation(B-rep) models.

2.2 Finite Element Mesh

In performing an analysis with FEA a continuum is discretised into an analytical model which consists of a mesh of points in the continuum (nodes) that are connected with elements. These elements have associated physical and material properties. There are also coordinate systems, groups (of nodes, elements and groups) and administrative data associated with the discretising mesh model. Load, constraint and output control data along with analysis selection information are combined with the discretising mesh model to form a complete input to an analysis. Once an analysis is performed, results data are output at the nodes and at one or more positions within an element. There may be other output data not associated with the mesh model such as eigenvalues or total strain energy.

This section looks at the analytical model, ie. the nodes and elements, together with their physical properties, coordinate systems and groups. The other data will be dealt with in the following schemas.

The following are within scope in this schema:

- submodels
- nodes
- coordinate space representations
- volume elements defined by a set of nodes
- volume elements defined by geometry
- element types
- degrees of freedom
- geometric property representations
- material representations
- groups
- units

Elements other than volume elements are out of scope for this version of the data model. They will be included in a later version.

2.3 FE Analysis Control

The FE Analysis Control data model covers all those areas of data which are needed by the FE code, in addition to the FE mesh, to perform an analysis. It describes the operations to be carried out upon the model as a set of analysis steps. A model may have one or more sets of control information. The control information also includes the administrative and configuration control information and the constraints upon the model which act for each analysis step.

The following are within scope in this schema:

- analysis selection and related information
- constraints
  - on single nodes
  - on sets of nodes
  - on geometric points
  - including real and imaginary parts
  - including time dependency
- loads
  - nodal
  - elemental (body force)
  - element face (pressure)
  - including all options as for constraints
- excitations not related to boundary conditions (e.g. currents in coils)
- output control information

2.4 Analysis type scope

The following types of analyses are within the scope of this data model:

- Linear static structural analysis of 3D continua.
- Plane stress
- Axisymmetric and simple plane strain
- Static, transient and steady state electromagnetic

3 Data Modelling Principles

All of the STEP data models were designed deliberately to be implementation independent. This results in models which can be used to generate either physical files or different types of database schemas. However, the schema which would be generated automatically may not be directly applicable to a specific type of database, or may result in inefficient use of that database. STEP includes a description of how to map from *EXPRESS* to a physical file but does not include a mapping from *EXPRESS* to database schemas.

When deriving the MIDAS data models every effort has been made to keep as close as possible to the available STEP models. However, there are restrictions in DEVA, the data base used within the project, as in almost any database, which do not allow the full *EXPRESS* language to be implemented without some modifications.

For example, the **SUB/SUPERTYPE** constructs cannot be implemented in DEVA. Where these occur in the STEP data models they have been removed and replaced by a single entity for each of the subtypes. The supertype has been replaced by a select type which allows other entities to reference any of the subtype entities. Where there are subtypes of subtypes forming a tree structure, entities have been included in the MIDAS model for each of the subtypes at the lower extremities of the structure. All the supertypes above this have been removed and their attributes inherited down to the subtype entities. This is in accordance with the STEP physical file mapping. There are a few entities where a slightly different approach has been taken and this has been explained in the data model together with the relevant **ENTITY** definition.

The other change which has been made to all entities is the removal of some of the strong typing. For example, many of the entities have an attribute **name** which has a type **label**. **label** is further defined to be of type **STRING**. This has been simplified in the MIDAS model so that the attribute **name** is of type **STRING**. This has been done to simplify the model and to aid implementation by those unfamiliar with STEP.

For each **ENTITY** in the data model the text gives the reference to the STEP entities on which that entity is based. This reference is given in terms of a reference to the relevant document as given in the bibliography and a section number in that document. To find out the names of the referenced entities a cross-reference table is given in Appendix A.

The text also gives a brief description of the purpose of the entity. This text is normally taken from the relevant STEP document unless the entity has been changed for the purpose of the MIDAS model. Some attribute definitions are give after the *EXPRESS* listing. For more details of the relevant entities and a full list of attribute definitions, reference should be made to the original STEP document.

The Finite Element Control part of the STEP data model uses the type **measure_or_unspecified_value** instead of **OPTIONAL** attributes. This results in the enumerated type **.UNSPECIFIED.** appearing in the file whenever a value is not specified. This has been replaced in the MIDAS data model by optional attributes.

4 Definitions, abbreviations and symbols

4.1 Terms defined in ISO 10303-1

This document makes use of the following terms defined in ISO 10303-1:

- application;
- data;
- data exchange;

4.2 Definitions and Abbreviations

For the purpose of this report, the following definitions and abbreviations apply:

arcwise connected an entity is arcwise connected if any two arbitrary points in its domain can be connected by a curve that lies entirely within the domain.

bounds the topological entities of lower dimensionality which mark the limits of a topological entity. The bounds of a face are loops and the bounds of an edge are vertices.

boundary representation solid model (B-rep) a type of geometric model in which the size and shape of the solid is defined in terms of the faces, edges and vertices which make up its boundary.

placement coordinate system a rectangular Cartesian coordinate system associated with the placement of a geometric entity in space, used to describe the interpretation of the attributes and to associate a unique parameterisation with curve and surface entities.

3d model A Finite Element Model that has geometry in three dimensions;

2d model A Finite Element Model that has geometry restricted to a plane that is either swept around an axis of symmetry (axisymmetric model) or projected perpendicular to a plane (plane model) to create the third dimension of the volume.

FEA Finite Element Analysis

EM Electromagnetic

DOF Degree of Freedom

5 Geometry

The following *EXPRESS* declaration begins the **midas_schema**.

EXPRESS specification:

```
*)
SCHEMA midas_schema;
(*
```

5.1 geometric_representation_item

Based on Part 42 [6] 4.4.4 and Part 43 [7] 4.4.2.

The subtypes have been replaced by select types and many of the subtypes have been omitted in this first version of the MIDAS data model. The select type may be extended to include more of the original subtypes at a later date.

EXPRESS specification:

```
*)
TYPE geometric_representation_item = SELECT
      (cartesian_point, fea_axis2_placement_3d,  manifold_solid_brep,
       facetted_brep);
END_TYPE;
(*
```

5.2 topological_representation_item

Based on Part 42 [6] 5.4.1 and Part 43 [7] 4.4.4.

The subtypes have been replaced by select types and many of the subtypes have been omitted in this first version of the MIDAS data model. The select type may be extended to include more of the original subtypes at a later date.

EXPRESS specification:

```
*)
TYPE topological_representation_item = SELECT
     (vertex_point, edge, oriented_edge, loop,
      face_bound, face, closed_shell);
END\_TYPE;
(*
```

5.3 representation_item

Based on Part 43 [7] 4.4.4.

The subtypes have been replaced by select types and many of the subtypes omitted in this first version of the MIDAS data model. The select type may be extended to include more of the original subtypes at a later date.

EXPRESS specification:

```
*)
TYPE representation_item = SELECT
     (geometric_representation_item,
      topological_representation_item);
END_TYPE;
(*
```

5.4 shape_representation

Based on Part 41 [5] 2.5.3.1 and Part 43 [7] 4.4.5.

A **shape_representation** is a specific kind (i.e. a **SUBTYPE**) of the STEP **SUPERTYPE** entity **representation** that represents a shape. All **geometric_representation_items** which are included as **items** in a **representation** having a **geometric_representation_context** are geometrically related. Any such **geometric_representation_item** is said to be geometrically founded in the context of that **representation**. No geometric relationship, such as distance between points, is assumed to exist for **geometric_representation_items** occurring as **items** in different **representations**.

EXPRESS specification:

```
*)
ENTITY shape_representation;
       name                 : STRING;
       representation_items : SET [1:?] of representation_item;
       context_of_items     : geometric_representation_context;
END_ENTITY;
(*
```

5.5 geometric_representation_context

Based on Part 42 [6] 4.4.1 and Part 43 [7] 4.4.5.

A **geometric_representation_context** is a distinct coordinate space, spatially unrelated to other coordinate spaces except as those coordinate spaces are specifically related by an appropriate transformation.

EXPRESS specification:

```
*)
ENTITY geometric_representation_context;
       context_identifier         : STRING;
       context_type               : STRING;
       coordinate_space_dimension : INTEGER;
END_ENTITY;
(*
```

5.6 cartesian_point

Based on Part 42 [6] 4.4.4, 4.4.3, 4.4.2 and Part 43 [7] 4.4.2.

A **cartesian_point** is a point defined by its coordinates in a rectangular Cartesian coordinate system, or in a parameter space. The entity is defined in a one, two or three-dimensional space as determined by the number of coordinates in the list.

EXPRESS specification:

```
*)
ENTITY cartesian_point;
       name            : STRING;
       coordinates     : LIST [1:3] OF REAL;
END_ENTITY;
(*
```

5.7 vertex_point

Based on Part 42 [6] 5.4.3, 5.4.3, 5.4.1, 4.4.2 and Part 43 [7] 4.4.4.

A **vertex_point** is a vertex that has its geometry defined as a point.

EXPRESS specification:

```
*)
ENTITY vertex_point;
       name            : STRING;
       vertex_geometry : cartesian_point;
END_ENTITY;
(*
```

5.8 edge

Based on Part 42 [6] 5.4.4, 5.4.1 and Part 43 [7] 4.4.4.

A **edge** is the topological construct corresponding to the connection between two vertices.

EXPRESS specification:

```
*)
ENTITY edge;
       name            : STRING;
       edge_start      : vertex_point;
       edge_end        : vertex_point;
END_ENTITY;
(*
```

5.9 oriented_edge

Based on Part 42 [6] 5.4.4, 5.4.1 and Part 43 [7] 4.4.4.

A **oriented_edge** is an **edge** constructed from another **edge** and contains a BOOLEAN orientation flag to indicate whether or not the orientation of the constructed **edge** agrees with the orientation of the original **edge**. Except for possible re-orientation, the **oriented_edge** is equivalent to the original **edge**.

EXPRESS specification:

```
*)
ENTITY oriented_edge;
       edge_element    : edge;
       orientation     : BOOLEAN;
END_ENTITY;
(*
```

Attribute definitions:

orientation: If TRUE, the topological orientation as used coincides with the orientation, from start vertex to end vertex, as the edge_element.

5.10 loop

This SELECT type has been created because of the removal of the SUPERTYPE construct between **edge_loop** and **poly_loop**.

EXPRESS specification:

```
*)
TYPE loop = SELECT
     (edge_loop, poly_loop);
END_TYPE;
(*
```

5.11 poly_loop

Based on Part 42 [6] 5.4.13, 5.4.10, 5.4.1 and Part 43 [7] 4.4.4.

A **poly_loop** is a loop with straight edges bounding a planar region in space. A **poly_loop** is a **loop** of genus 1 where the loop is represented by an ordered coplanar collection of **points** forming the vertices of the loop. The loop is composed of straight line segments joining a point in the colletion to the succeeding point in the collection. The closing segment is from the last to the first point in the collection. The direction of the loop is in the direction of the line segments.

EXPRESS specification:

```
*)
ENTITY poly_loop;
       name          : STRING;
       polygon       : LIST [3:?] OF UNIQUE cartesian_point;
END_ENTITY;
(*
```

5.12 edge_loop

Based on Part 42 [6] 5.4.12, 5.4.7, 5.4.10, 5.4.1 and Part 43 [7] 4.4.4

An **edge_loop** is a **loop** with nonzero extent. It is a path in which the start and end vertices are the same.

EXPRESS specification:

```
*)
ENTITY edge_loop;
       name          : STRING;
       edge_list     : LIST [1:?] OF UNIQUE oriented_edge;
END_ENTITY;
(*
```

5.13 face_bound

Based on Part 42 [6] 5.4.14, 5.4.1 and Part 43 [7] 4.4.4.

A **face_bound** is a loop which is intended to be used for bounding a face.

EXPRESS specification:

```
*)
ENTITY face_bound;
       name          : STRING;
       bound         : loop;
       orientation   : BOOLEAN;
END_ENTITY;
(*
```

5.14 face

Based on Part 42 [6] 5.4.16, 5.4.1 and Part 43 [7] 4.4.4.

A **face** is a topological entity of dimensionality 2 corresponding to the intuitive notion of a piece of a surface bounded by loops. The topological normal n is associated with the face, such that the cross product nxt points toward the interior of the face where t is the tangent to a bounding loop. That is, each loop runs counter-clockwise around the face when viewed from above, if we consider the normal n to point up.

EXPRESS specification:

```
*)
ENTITY face;
       name            : STRING;
       bounds          : SET [1:?] OF face_bound;
END_ENTITY;
(*
```

5.15 closed_shell

Based on Part 42 [6] 5.4.25, 5.4.20, 5.4.1 and Part 43 [7] 4.4.4.

A **closed_shell** is a shell of dimensionality 2 which typically serves as a bound for a region. The topological normal of the shell is defined as being directed from the finite to the infinite region. The shell is defined by a collection of **faces**. The sense of each **face** shall agree with the shell normal as defined above.

EXPRESS specification:

```
*)
ENTITY closed_shell;
       name            : STRING;
       cfs_faces       : SET [1:?] OF face;
END_ENTITY;
(*
```

5.16 oriented_closed_shell

Based on Part 42 [6] 5.4.26, 5.4.25, 5.4.20, 5.4.1 and Part 43 [7] 4.4.4.

A **oriented_closed_shell** is a **closed_shell** constructed from another **closed_shell** and contains a BOOLEAN orientation flag to indicate whether or not the orientation of the constructed **closed_shell** agrees with the orientation of the original **closed_shell**.

EXPRESS specification:

```
*)
ENTITY oriented_closed_shell;
       closed_shell_element : closed_shell;
       orientation          : BOOLEAN;
END_ENTITY;
(*
```

5.17 manifold_solid_brep

Based on Part 42 [6] 6.4.2, 6.4.3, 6.4.1, 4.4.2 and Part 43 [7] 4.4.4. Changed to include the attribute **voids** rather than have an additional entity.

A **manifold_solid_brep** is a finite, arcwise connected volume bounded by one or more surfaces, each of which is a connected, oriented, finite, closed 2-manifold. There is no restriction on the number of through holes, nor on the number of voids within the volume.

The Boundary Representation (B-rep) of a manifold solid utilises a graph of edges and vertices embedded in a connected, oriented, finite, closed two manifold surface. The embedded graph divides the surface into arcwise connected areas known as faces. the edges and vertices, therefore, form the boundaries of the faces and the domain of a face does not inlcude its boundaries. The embedded graph may be disconnected and may be a pseudograph. The graph is labelled; that is, each entity in the graph has a unique identity. The geometric surface definition used to specify the geometry of a face shall be 2-manifold embeddable in the plane within the domain of the face. In other words, it shall be connected, oriented, finite, non-self-intersecting and of surface genus 0.

Faces do not intersect except along their boundaries. Each edge along the boundary of a face is shared by at most one other face in the assemblage. The assemblage of edges in the B-rep do not intersect except at their boundaries (i.e., vertices). The geometric curve definition used to specify the geometry of an edge shall be arcwise connected and shall not self intersect or overlap within the domain of the edge. The geometry of an edge shall be consistent with the geometry of the facs of which it forms a partial bound.

The geomtry used to define a vertex shall be consistent with the geometry of the faces and edges of which it forms a partial bound.

In the STEP model a **brep_with_voids** is a special **subtype** of the **manifold_solid_brep** which contains one or more voids in its interior. This is handled in the MIDAS model by the attribute **voids**. The voids are represented by **oriented_closed_shells** which are defined so that the **oriented_closed_shell** normals point into the void, that is, with **orientation** FALSE.

A **manifold_solid_brep** shall not reference **poly_loops**.

entity.

EXPRESS specification:

```
*)
ENTITY manifold_solid_brep;
       name              : STRING;
       outer             : closed_shell;
       voids             : LIST [0:?] OF oriented_closed_shell;
END_ENTITY;
(*
```

5.18 facetted_brep

Based on Part 42 [6] 6.4.4, 6.4.3, 6.4.2, 6.4.1, 4.4.2 and Part 43 [7] 4.4.4. Changed to include the attribute **voids** rather than have an additional entity as for **manifold_solid_brep**.

A **faceted_brep** is a simple form of boundary representation model in which all faces are planar and all edges are straight lines. Unlike the B-rep model, edges and vertices are not represented explicitly in the model but are implicitly available through the **poly_loop** entity. All the bounding loops of all the faces of all the shells in the **facetted_brep** shall be of type **poly_loop**.

EXPRESS specification:

```
*)
ENTITY facetted_brep;
       name              : STRING ;
       outer             : closed_shell;
       voids             : LIST [0:?] OF closed_shell;
END_ENTITY;
(*
```

6 Data Model for Finite Element Model

A finite element analysis model is a collection of information that represents the finite element analysis of a product. This information includes nodes, elements, materials, properties and groups which are combined to form a discrete mesh model of the product.

6.1 element_order

From Part 104 [3] 5.3.3.

EXPRESS specification:

```
*)
TYPE element_order = ENUMERATION OF
     (linear, quadratic, cubic);
END_TYPE;
(*
```

Enumerated item definitions:

linear: The element basic interpolation order is linear

quadratic: The element basic interpolation order is quadratic

cubic: The element basic interpolation order is cubic

6.2 matrix_property_type

From Part 104 [3] 5.3.10.

EXPRESS specification:

```
*)
TYPE matrix_property_type = ENUMERATION OF
     (stiffness, mass);
END_TYPE;
(*
```

6.3 volume_variable

Based on Part 104 [3] 6.3.14. User definied variables have been removed.

A **volume_variable** is a field variable within the volume of an element.

EXPRESS specification:

```
*)
TYPE volume_variable = SELECT
     (volume_scalar_variable, volume_angular_variable,
      volume_vector_3d_variable, volume_tensor2_3d_variable);
END_TYPE;
(*
```

6.4 volume_scalar_variable

Based on Part 104 [3] 6.3.21. User definied variables have been removed and additional keywords needed for electromagnetic analysis added.

A **volume_scalar_variable** is a scalar field variable that is evaluated at a point within the volume of an element.

EXPRESS specification:

```
*)
TYPE volume_scalar_variable = ENUMERATION OF
     (temperature, moisture, reference_temperature,
      strain_energy_per_unit_volume, electric_scalar_potential,
      magnetic_scalar_potential, electric_charge_density);
END_TYPE;
(*
```

6.5 volume_angular_variable

Based on Part 104 [3] 6.3.25. User definied variables have been removed.

A **volume_angular_variable** is a scalar variable that is applied to a volume about the origin of the founding placement of the using entity, and about the Z axis of that placement.

EXPRESS specification:

```
*)
TYPE volume_angular_variable = ENUMERATION OF
     (constant_angular_acceleration);
END_TYPE;
(*
```

6.6 volume_vector_3d_variable

Based on Part 104 [3] 6.3.33. Additional keywords needed for electromagnetic analysis have been added.

A **volume_vector_3d_variable** is a three dimensional vector field variable at a point within the volume of an element.

EXPRESS specification:

```
*)
TYPE volume_vector_3d_variable = ENUMERATION OF
     (position, applied_force_per_unit_volume,
      applied_moment_per_unit_volume,
      displacement, infinitesimal_rotation, acceleration,
      magnetic_vector_potential, electric_field_intensity,
      magnetric_field_intensity, electric_flux_density,
      magnetic_flux_density, electric_current_density);
END_TYPE;
(*
```

6.7 volume_tensor2_3d_variable

From Part 104 [3] 6.3.40.

A **volume_tensor2_3d_variable** is a three dimensional second order tensor field variable, that is evaluated at a point within the volume of an element.

EXPRESS specification:

```
*)
TYPE volume_tensor2_3d_variable = ENUMERATION OF
     (elastic_strain, stress);
END_TYPE;
(*
```

6.8 degree_of_freedom

Based on Part 104 [3] 5.3.15. Additional types of degrees of freedom specific to electromagnetic analysis added.

EXPRESS specification:

```
*)
TYPE degree_of_freedom = ENUMERATION OF
     (x_translation, y_translation, z_translation,
      x_rotation, y_rotation, z_rotation, warp,
      electric_scalar_potential,magnetic_scalar_potential,
      magnetic_vector_potential,electric_field_intensity,
      magnetic_field_intensity, electric_flux_density,
      magnetic_flux_density, electric_charge_density,
      electric_current_density);
END_TYPE;
(*
```

6.9 integration_rule

From Part 104 [3] 5.3.16.

EXPRESS specification:

```
*)
TYPE integration_rule = ENUMERATION OF
     (gaussian, simpson);
END_TYPE;
(*
```

6.10 shape_function

Based on Part 104 [3] 5.3.17. Type **unknown_shape_function** has been removed as agreed a Greenville meeting.

EXPRESS specification:

```
*)
TYPE shape_function = ENUMERATION OF
     (lagrangian, serendipity, hermitian, unspecified);
END_TYPE;
(*
```

6.11 volume_2d_element_representation

From Part 104 [3] 5.3.20.

EXPRESS specification:

```
*)
TYPE volume_2d_element_representation = SELECT
     (axisymmetric_volume_2d_element_representation,
      plane_volume_2d_element_representation);
END_TYPE;
(*
```

6.12 fea_axis2_placement_3d

Based on Part 104 [3] 5.5.2, Part 42 [6] 4.4.15, 4.4.12, 4.4.2 and Part 43 [7] 4.4.4.

A **fea_axis2_placement_3d** is a location and orientation in three dimensional space of two mutually perpendicular axes, and declares that the placement coordinate system may be either Cartesian, Cylindrical or Spherical. All coordinate systems in a finite element analysis model should be right handed.

EXPRESS specification:

```
*)
ENTITY fea_axis2_placement_3d;
       name                    : STRING;
       location                : cartesian_point;
       axis                    : direction;
       ref_direction           : direction;
       system_type             : coordinate_system_type;
       system_id               : STRING;
       description             : STRING;
END_ENTITY;
(*
```

Attribute definitions:

location: The geometric position of a reference point, such as the centre of a circle, of the item to be located.

axis: The exact direction of the local Z axis.

ref_direction: The direction used to determine the direction of the local X axis. If necessary an adjustment is made to maintain orthogonality to the **axis** direction.

system_type: The type of placement coordinate system, which may be a right handed Cartesian, Cylindrical or Spherical coordinate system.

system_id: An application defined identifier for the coordinate system, which is unique within the fea_model.

description: Additional information about the formulation or purpose of the coordinate system.

6.13 direction

Based on Part 42 [6] 4.4.10, 4.4.2 and Part 43 [7] 4.4.4.

This entity defines a general direction vector in two or three dimensional space. The actual magnitudes of the components have no effect upon the direction being defined, only the ratios $x : y : z$ or $x : y$ are significant.

EXPRESS specification:

```
*)
ENTITY direction;
       name                      : STRING;
       direction_ratios          : LIST [2:3] OF REAL;
END_ENTITY;
(*
```

6.14 coordinate_system_type

From Part 104 [3] 5.3.2.

EXPRESS specification:

```
*)
TYPE coordinate_system_type = ENUMERATION OF
     (cartesian, cylindrical, spherical);
END_TYPE;
(*
```

6.15 fea_model

Based on Part 104 [3] 5.4.1.

EXPRESS specification:

```
*)
TYPE fea_model = SELECT
     (fea_model_3d, fea_model_2d);
END_TYPE;
(*
```

6.16 fea_model_3d

Based on Part 104 [3] 5.4.1 and 5.4.3, and Part 43 [7] 4.4.5.

The original **fea_model** entities were a **SUBTYPE** of representation and as such have an attribute **items** which points to a set of **representation_items**. Unfortunately, without knowing what entities in the data model are **SUBTYPES** of **representation_item** it is difficult to tell what this attribute can point to and, indeed, which entities should be included in this set. This mechanism has been replaced by adding in extra attributes with direct references to ease understanding and implementation.

The **response_property** attribute in the STEP model connects to a product and hence its geometry in a very long-winded way. For the first version of the MIDAS model this has been replaced by a direct pointer to a named geometry.

EXPRESS specification:

```
*)
ENTITY fea_model_3d;
       name                    : STRING;
       master_coordinate_system: fea_axis2_placement_3d;
       global_units            : global_unit_assigned_context;
       creating_software       : STRING;
       intended_analysis_code  : STRING;
       description             : STRING;
       analysis_type           : STRING;
       response_property       : OPTIONAL shape_representation;
UNIQUE
        UR1: name;
END_ENTITY;
(*
```

Attribute definitions:

Response_property: is the definitional shape aspect of a model.

6.17 fea_model_2d

From Part 104 [3] 5.4.1 and 5.4.2, and Part 43 [7] 4.4.5.

EXPRESS specification:

```
*)
ENTITY fea_model_2d;
       name                    : STRING;
       master_coordinate_system: fea_axis2_placement_3d;
       global_units            : global_unit_assigned_context;
       creating_software       : STRING;
       intended_analysis_code  : STRING;
       description             : STRING;
       analysis_type           : STRING;
       response_property       : OPTIONAL shape_representation;
       type_of_2d_analysis     : axi_or_plane;
UNIQUE
        UR1: name;
END_ENTITY;
(*
```

Attribute definitions:

response_property: is the definitional shape aspect of a model.

6.18 axi_or_plane

From Part 104 [3] 5.3.1.

An **axi_or_plane** specifies whether a **fea_2d_model** is axisymmetric or plane.

EXPRESS specification:

```
*)
TYPE axi_or_plane = ENUMERATION OF
     (axisymmetric, planar);
END_TYPE;
(*
```

Enumerated item definitions:

axisymmetric: The **fea_model** is an axisymmetric analysis model where two dimensional element geometry is assumed to be swept about the j axis of the founding coordinate system to create a volume.

planar: The **fea_model** is a two dimensional analysis model where two dimensional elements are assumed to be extruded perpendicular to the analysis plane to create a volume.

6.19 global_unit_assigned_context

Based on Part 41 [5] 4.14.4.20, Part 43 [7] 4.4.2.

A **global_unit_assigned_context** is a **representation_context** in which the units apply to all **measure_values** of the correct kind.

EXPRESS specification:

```
*)
ENTITY global_unit_assigned_context;
       context_identifier       : STRING;
       context_type             : STRING;
       units                    : SET [1:?] OF unit;
END_ENTITY;
(*
```

Attribute definitions:

units: The units which apply in this context ie. throughout the model.

6.20 unit

From Part 41 [5] 4.14.3.22.

EXPRESS specification:

```
*)
TYPE unit = SELECT
     (named_unit, derived_unit);
END_TYPE;
(*
```

6.21 named_unit

Based on Part 41 [5] 4.14.4.1.

This entity in Part 41 is a complex **SUPERTYPE** entity. It has three subtypes, **si_unit**, **conversion_based_unit** and **context_dependent_unit**. Only the first two of these have been implemented in this version of the MIDAS data model. The select type may be extended at a later date.

The **unit** entity is also a supertype of 12 subtype entities which define the type of the unit e.g. **length_unit**. This has been implemented by adding an extra attribute **unit_type** to the above two subtype entities.

EXPRESS specification:

```
*)
TYPE named_unit = SELECT
    (si_unit, conversion_based_unit);
END_TYPE;
(*
```

6.22 derived_unit

From Part 41 [5] 4.14.4.19.

A **derived_unit** is an expression of units. For example, Newtons per square millimetre is a **derived_unit**.

EXPRESS specification:

```
*)
ENTITY derived_unit;
       elements                 : SET [1:?] OF derived_unit_element;
END_ENTITY;
(*
```

Attribute definitions:

elements: The group of units and their exponents that define the **derived_unit**.

6.23 derived_unit_element

From Part 41 [5] 4.14.4.18.

A **derived_unit_element** is one of the unit quantities which makes up a **derived_unit**. For example, Newtons per square millimetre is a derived unit. It has two elements, Newton whose exponent has a value of 1 and millimetre whose exponent is -2.

EXPRESS specification:

```
*)
ENTITY derived_unit_element;
       unit                     : named_unit;
       exponent                 : REAL;
END_ENTITY;
(*
```

Attribute definitions:

unit: The fixed quantity which is used as the mathematical factor

exponent: the power that is applied to the **unit** attribute

6.24 si_unit

Based on Part 41 [5] 4.14.4.2 and 4.14.4.1.

An **si_unit** is the fixed quantity used as a standard in terms of which items are measured as defined by ISO 1000 (clause 2).

EXPRESS specification:

```
*)
ENTITY si_unit;
       dimensions               : dimensional_exponents;
       unit_type                : unit_type;
       prefix                   : OPTIONAL si_prefix;
       name                     : si_unit_name;
END_ENTITY;
(*
```

Attribute definitions:

dimensions: The exponents of the base properties by which the **named_unit** is defined.

6.25 conversion_based_unit

Based on Part 41 [5] 4.14.4.3 and 4.14.4.1.

A **conversion_based_unit** is a unit that is defined based on a **measure_with_unit**. For example, an inch is a converted unit. It is from the Imperial system, its name is "inch" and it can be related to the **si_unit**, millimetre, through a **measure_with_unit** whose value is 25.4 millimetre.

EXPRESS specification:

```
*)
ENTITY conversion_based_unit;
       dimensions               : dimensional_exponents;
       unit_type                : unit_type;
       name                     : STRING;
       conversion_factor        : measure_with_unit;
END_ENTITY;
(*
```

Attribute definitions:

dimensions: The exponents of the base properties by which the **named_unit** is defined.

name: The word or group of words by which the **conversion_based_unit** is referred to.

conversion_factor: The physical quantity from which the **converted_unit** is derived.

6.26 si_unit_name

From Part 41 [5] 4.14.3.23.

An **si_unit_name** is the name of an SI unit. The definitions of the names of SI units are specified in ISO 1000 (clause 2).

<u>*EXPRESS* specification:</u>

```
*)
TYPE si_unit_name = ENUMERATION OF
     (metre, gram, second, ampere, kelvin, mole, candela, radian,
      steradian, hertz, newton, pascal, joule, watt, coulomb, volt,
      farad, ohm, siemens, weber, tesla, henry, degree_Celsius, lumen,
      lux, becquerel, gray, sievert);
END_TYPE;
(*
```

6.27 si_prefix

From Part 41 [5] 4.14.3.24.

An **si_prefix** is the name of a prefix that may be associated with an **si_unit**. The definitions of SI prefixes are specified in ISO 1000 (clause 2).

<u>*EXPRESS* specification:</u>

```
*)
TYPE si_prefix = ENUMERATION OF
     (exa, peta, tera, giga, mega, kilo, hecto, deca, deci,
      centi, milli, micro, nano, pico, femto, atto);
END_TYPE;
(*
```

6.28 unit_type

Based on Part 104 [3] 4.14.4.1.

This type has been created out of the subtypes of **named_unit** to remove the necessity for a proliferation of entities.

<u>*EXPRESS* specification:</u>

```
*)
TYPE unit_type = ENUMERATION OF
     (length_unit, mass_unit, time_unit, electric_current_unit,
      thermodynamic_temperature_unit, amount_of_substance_unit,
      luminous_intensity_unit, plane_angle_unit, solid_angle_unit,
      area_unit, volume_unit, ratio_unit);
END_TYPE;
(*
```

6.29 dimensional_exponents

From Part 41 [5] 4.14.4.17.

The dimensionality of any quantity can be expressed as a product of powers of the dimensions of base quantities. The **dimensional_exponents** entity defines the powers of the dimensions of the base quantities. All the physical quantities are founded on seven base quantities (ISO 31 (clause 2).

EXPRESS specification:

```
*)
ENTITY dimensional_exponents;
       length_exponent                    : REAL;
       mass_exponent                      : REAL;
       time_exponent                      : REAL;
       electric_current_exponent          : REAL;
       thermodynamic_temperature_exponent : REAL;
       amount_of_substance_exponent       : REAL;
       luminous_intensity_exponent        : REAL;
END_ENTITY;
(*
```

Attribute definitions:

length_exponent: The power of the length base quantity.

etc.:

6.30 measure_with_unit

Based on Part 41 [5] 4.14.4.21 and 4.14.3.1. The strong typing for the attribute **value_component** has been removed and an additional attibute **unit_type** has been added to remove the need for multiple entities to cope with the numerous subtypes.

A **measure_with_unit** is the specification of a physical quantity as defined in ISO 31 (clause 2).

EXPRESS specification:

```
*)
ENTITY measure_with_unit;
       unit_type              : unit_type;
       value_component        : REAL;
       unit_component         : unit;
END_ENTITY;
(*
```

Attribute definitions:

value_component: The value of the physical quantity when expressed in the specified units.

unit_component: The **unit** in which the physical quantity is expressed.

6.31 node

Based on Part 104 [3] 5.6.3 and 5.6.1 and Part 43 [7] 4.4.5. Takes account of changes made at ISO meeting in Greenville.

A node is a discretisation point for the field variables of the finite element analysis model. The connection to **point** in the STEP model is from the fact that node is a **SUBTYPE** of **representation** which points to a **SET** of **representation_items**. The **ENTITY point** is a **SUBTYPE** of **geometric_representation_item** which is in turn a **SUBTYPE** of **representation_item**. This mechanism has been replaced in the MIDAS model by a direct attribute in **node** pointing to **point**.

<u>*EXPRESS* specification</u>:

```
*)
ENTITY node;
       name                    : STRING;
       model_ref               : fea_model;
       point                   : cartesian_point;
UNIQUE
  UR1 : model_ref, name;
END_ENTITY;
(*
```

<u>Attribute definitions</u>:

name: A unique application defined identifier of a node.

model_ref: An application defined identifier of the **fea_model** which possesses the node.

6.32 element_representation

Based on Part 104 [3] 5.7.1, Part 43 [7] 4.4.12 and 4.4.5.

An **element_representation** is the aspect of a finite element which represents the mathematical relationships between the nodes of a finite element model. Only volume elements have been included in this first version of the data model. The select type can be extended at a later date to include more element types. The select type replaces the original **SUB/SUPERTYPE** construct.

<u>*EXPRESS* specification</u>:

```
*)
TYPE element_representation = SELECT
      (volume_3d_element_representation,
       volume_2d_element_representation);
END_TYPE;
(*
```

6.33 volume_3d_element_representation

Based on Part 43 [7] 4.4.12, 4.4.5, Part 104 [3] 5.7.1, 5.7.3 and 5.7.4.

A **volume_3d_element_representation** is a three dimensional shape. Similarly to the **fea_model** entities, the method of referencing a coordinate system via the route of a representation and set of representation items, has been changed to a direct attribute reference. Also the reference to geometry was achieved in the STEP model by a different **SUBTYPE** , **element_shape_representation** which points to a **structural_response_property**, this has been changed to an **OPTIONAL** attribute pointing directly to a **shape_representation.**

Higher order elements are also achieved in the STEP model with an additional **SUBTYPE** entity which has been replaced by an **OPTIONAL** attribute.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_representation;
       name                        : STRING;
       required_node_list          : OPTIONAL LIST [1:?] OF node;
       model_ref                   : fea_model_3d;
       element_descriptor          : volume_3d_element_descriptor;
       material_properties         : SET [1:?] OF fea_material_representation;
       additional_node_list        : OPTIONAL LIST [1:?] OF node;
       material_coordinate_system : volume_3d_element_coordinate_system;
       response_property           : OPTIONAL shape_representation;
UNIQUE
  UR1 : model_ref, name;
END_ENTITY;
(*
```

Attribute definitions:

required_node_list: The list of nodes which are essential for the element. These are the vertex nodes. This attribute should be be omitted if and only if the type of element shape in the element_descriptor is geometric, i.e. if the element is defined by its geometry.

element_descriptor: The collection of information that specifies a **volume_3d_element_representation.**

6.34 volume_3d_element_coordinate_system

From Part 104 [3] 5.3.26.

A **volume_3d_element_coordinate_system** is the orthogonal coordinate system for a **volume_3D_element** that shall be either an arbitrary of parametric coordinate system.

EXPRESS specification:

```
*)
TYPE volume_3d_element_coordinate_system = SELECT
      (arbitrary_volume_3d_element_coordinate_system,
       parametric_volume_3d_element_coordinate_system);
END_TYPE;
(*
```

6.35 arbitrary_volume_3d_element_coordinate_system

Based on Part 104 [3] 5.9.2, 5.4.5, and Part 43 [7] 4.4.4.

An **arbitrary_volume_3d_element_coordinate_system** is an arbitrary orthogonal coordinate system for a **volume_3d_element.**

EXPRESS specification:

```
*)
ENTITY arbitrary_volume_3d_element_coordinate_system;
       name                    : STRING;
       coordinate_system       : fea_axis2_placement_3d;
END_ENTITY;
(*
```

Attribute definitions:

coordinate_system: At a point within the element information is defined with respect to the local orthogonal coordinate system triad of the specified coordinate system at that point.

6.36 parametric_volume_3d_element_coordinate_system

Based on Part 104 [3] 5.9.3, 5.9.18, 5.4.5, and Part 43 [7] 4.4.4

A **parametric_volume_3d_element_coordinate_system** is the orthogonal coordinate system for a **volume_3d_element**. At each point in the element an intermediate orthogonal coordinate system is derived from the parametric coordinate system of an element.

EXPRESS specification:

```
*)
ENTITY parametric_volume_3d_element_coordinate_system;
       name                    : STRING;
       axis_1                  : INTEGER;
       axis_2                  : INTEGER;
       angles                  : ARRAY [1:3] OF REAL;
END_ENTITY;
(*
```

Attribute definitions:

axis_1: The first parametric axis used to derive the orthogonal coordinate system.

axis_2: The second parametric axis used to derive the orthogonal coordinate system.

angles: Three Euler angles that define the orthogonal element coordinate system.

6.37 volume_3d_element_descriptor

Based on Part 43 [7] 4.4.4 , Part 104 [3] 5.4.5 , 5.7.13 and 5.7.14.

A **volume_3d_element_descriptor** is a collection of information that specifies a volume 3D element.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_descriptor;
       name                    : STRING;
       topology_order          : element_order;
       description             : STRING;
       purpose                 : SET [1:?] OF volume_element_purpose;
       shape                   : volume_3d_element_shape;
END_ENTITY;
(*
```

Attribute definitions:

topology_order: the highest degree polynomial interpolation function used to describe the geomtric shape of any edge of an element, which in turn is used to relate the additional node list of the element to the appropriate toplogy diagram.

description: Additional information about the formulation or purpose of an element.

purpose: The enumerated value specifying the response of a **volume_3d_element_representation**.

shape: The geometric shape of a **volume_3d_element_representation**.

6.38 volume_element_purpose

Based on Part 104 [3] 5.3.5. Additional electromagnetic keywords.

This is an enumeration of the type of strain-displacement or heat-transfer relationship assumed for a volume element.

EXPRESS specification:

```
*)
TYPE volume_element_purpose = ENUMERATION OF
     (stress_displacement, electostatic,
      magnetostatic, electrodynamic,
      electromagnetic);
END_TYPE;
(*
```

Enumerated item definitions:

stress_displacement: indicates that the element does not consider specialised behaviour outside the continuum analysis domain such as crack tip analyses.

6.39 volume_3d_element_shape

Based on Part 104 [3] 5.3.8. Additional **geometric** keyword to allow for the definition of the element to depend upon its geometry.

An enumeration of the shape for a three dimensional finite element.

EXPRESS specification:

```
*)
TYPE volume_3d_element_shape = ENUMERATION OF
     (hexahedron,
      wedge,
      tetrahedron,
      pyramid,
      geometric);
END_TYPE;
(*
```

Enumerated item definitions:

geometric: indicates that no node list will be given in the element representation and the shape of the element will be determined from the geometry associated with it via the attribute **response_property** which points to a **shape_representation** entity.

6.40 volume_3d_element_basis

From Part 104 [3] 5.7.17.

A **volume_3d_element_basis** is the information that forms the basis of a **volume_3d_element**.

EXPRESS specification:

```
*)
ENTITY volume_element_basis;
       descriptor               : volume_3d_element_descriptor;
       variable                 : volume_variable;
       cariable_order           : element_order;
       variable_shape_function  : shape_function;
       evaluation_points        : LIST [1:?] OF volume_element_location;
END_ENTITY;
(*
```

Attribute definitions:

descriptor: The association to the information describing a **volume_3d_element_representation.**

variable: The variable to be associated with an order and shape function for a **volume_3d_element_representation.**

variable_order: The mathematical order of the polynomials used to define the variable shape function.

variable_shape_function: The type of polynomial shape function used to interpolate the variable within a **volume_3d_element_representation.**

evaluation_points: The locations within a **volume_3d_element_representation** where the variable is evaluated using the **variable_shape_function.**

6.41 volume_3d_element_integrated_matrix

Based on Part 104 [3] 5.10.6, 5.10.7 and 5.4.5.

A **volume_3d_element_integrated_matrix** is the matrix to be integrated for a volume 3D element and the method of integration.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_integrated_matrix;
       name                     : STRING;
       descriptor               : volume_3d_element_descriptor;
       matrix_property_type     : matrix_property_type;
```

```
        integration_description : STRING;
        integration_definition  : volume_3d_element_field_integration;
END_ENTITY;
(*
```

Attribute definitions:

descriptor: The association to the information describing a **volume_element_representation.**

matrix_property_type: The type of matrix being evaluated

integration_description: The interpolation rule and integration method.

integration_definition: A definition of the integration within the 3D volume.

6.42 volume_3d_element_field_integration

From Part 104 [3] 5.10.8.

A **volume_3d_element_field_integration** is a three dimensional volume field integration which shall be either algebraic, by rule, or explicit.

EXPRESS specification:

```
*)
TYPE volume_3d_element_field_integration = SELECT
     (element_integration_algebraic,
      volume_3d_element_field_integration_rule,
      volume_3d_element_field_integration_explicit);
END_TYPE;
(*
```

6.43 element_integration_algebraic

From Part 104[3] 5.10.9.

An **element_integration_algebraic** is an element integration that is exact; therefore, no numerical integration information is required.

EXPRESS specification:

```
*)
TYPE element_integration_algebraic = ENUMERATION OF
    (algebraic);
END_TYPE;
(*
```

6.44 volume_3d_element_field_integration_rule

From Part 104 [3] 5.10.10.

A **volume_3d_element_field_integration_rule** is the integration rule and order for a volume 3D element.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_field_integration_rule;
       integration_rule        : integration_rule;
       integration_order       : ARRAY [1:3] OF INTEGER;
END_ENTITY;
(*
```

Attribute definitions:

integration_rule: The integration rule for the quantity being integrated.

integration_order: The order of the specified rule for the quantity being integrated. A separate integration order is specified for each parametric axis direction established graphically in the sequence (ξ, η, ζ).

6.45 volume_3d_element_field_integration_explicit

From Part 104 [3] 5.10.11.

A **volume_3d_element_field_integration_explicit** is the explicit numerical integration for a volume 3D element.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_field_integration_explicit;
       integration_positions_and_weights : SET [1:?] OF volume_position_weight;
END_ENTITY;
(*
```

Attribute definitions:

integration_positions_and_weights: The integration positions for the quantity being integrated, and the corresponding weights for each integration position.

6.46 volume_position_weight

From Part 104 [3] 5.10.12.

A **volume_position_weight** is an integration position within a volume element, and its weighting factor.

EXPRESS specification:

```
*)
ENTITY volume_position_weight;
       integration_position    : volume_element_location;
       integration_weight      : REAL;
END_ENTITY;
(*
```

Attribute definitions:

integration_position: The integration position for the quantity being integrated.

integration_weight: The weight for the integration position.

6.47 volume_element_location

From Part 104 [3] 5.11.2.

A **volume_element_location** is a location within a volume element, and specifies its parametric coordinates (ξ, η and ζ).

EXPRESS specification:

```
*)
ENTITY volume_element_location;
       coordinates              : fea_parametric_point;
END_ENTITY;
(*
```

Attribute definitions:

coordinates: The coordinates of the location.

6.48 fea_parametric_point

Based on Part 104 [3] 5.11.1, Part 41 [5] 4.4.3 and 4.4.2.

A **fea_parametric_point** is a position within a finite element, and specifies its parametric coordinates, (ξ, η and ζ).

EXPRESS specification:

```
*)
ENTITY fea_parametric_point;
       name                     : STRING;
       coordinates              : LIST [1:3] OF REAL;
END_ENTITY;
(*
```

Attribute definitions:

coordinates: The coordinates of the position. The first value of the coordinate is the ξ coordinate, the second the η coordinate, and the third the ζ coordinate (if applicable).

6.49 volume_2d_element_descriptor

From Part 104 [3] 5.3.23

EXPRESS specification:

```
*)
TYPE volume_2d_element_descriptor = SELECT
      (axisymmetric_volume_2d_element_descriptor,
       plane_volume_2d_element_descriptor);
END_TYPE;
(*
```

6.50 axisymmetric_volume_2d_element_representation

Based on Part 43 [7] 4.4.12, 4.4.5, Part 104 [3] 5.7.1, 5.7.3 and 5.7.5.

An **axisymmetric_volume_2d_element_representation** is a two dimensional shape swept about the j axis of the (i, j) plane of the founding coordinate system. An **axisymmetric_volume_2d_element_representation** shall lie in the (i, j) plane of the founding coordinate system.

Other changes have been made as for **volume_3d_element_representation**.

EXPRESS specification:

```
*)
ENTITY axisymmetric_volume_2d_element_representation;
       name                      : STRING;
       required_node_list        : OPTIONAL LIST [1:?] OF node;
       model_ref                 : fea_model_2d;
       element_descriptor        : axisymmetric_volume_2d_element_descriptor;
       material_properties       : SET [1:?] of material_property_representation;
       element_properties        : axisymmetric_2d_element_property;
       additional_node_list      : OPTIONAL LIST [1:?] OF node;
       material_coordinate_sys   : volume_2d_element_coordinate_system;
       response_property         : OPTIONAL shape_representation;
UNIQUE
  UR1 : model_ref, name;
END_ENTITY;
(*
```

Attribute definitions:

model_ref: An application defined identifier of the **fea_model_2d** which possesses the volume 3d element.

6.51 volume_2d_element_coordinate_system

Part 104 [3] 5.3.27.

A **volume_2d_element_coordinate_system** is the orthogonal coordinate system for a **volume_2d_element** that shall be either an arbitrary or parametric coordinate system.

EXPRESS specification:

```
*)
TYPE volume_2d_element_coordinate_system = SELECT
     (arbitrary_volume_2d_element_coordinate_system,
      parametric_volume_2d_element_coordinate_system);
END_TYPE;
(*
```

6.52 arbitrary_volume_2d_element_coordinate_system

Based on Part 104 [3] 5.9.4, 5.4.5, and Part 43 [7] 4.4.4.

An **arbitrary_volume_2d_element_coordinate_system** specifies an arbitrary orthogonal coordinate system for a volume two dimensional element, for which the x axis shall be normal to the 2D analysis plane and in the direction of the k axis of the 2D analysis plane definition coordinate system.

EXPRESS specification:

```
*)
ENTITY arbitrary_volume_2d_element_coordinate_system;
       name                    : STRING;
       orientation             : direction;
END_ENTITY;
(*
```

Attribute definitions:

direction: The direction used to orient the orthogonal coordinate system at each point within the element.

6.53 parametric_volume_2d_element_coordinate_system

A **parametric_volume_2d_element_coordinate_system** is an orthogonal coordinate system for a volume 2D element. The full explanation is given in an extract taken from Part 104 given in Appendix G.

Based on Part 104 [3] 5.9.5, 5.9.18, 5.4.5, and Part 43 [7] 4.4.4

EXPRESS specification:

```
*)
ENTITY parametric_volume_2d_element_coordinate_system;
        name                   : STRING;
        axis                   : INTEGER;
        angle                  : REAL;
END_ENTITY;
(*
```

Attribute definitions:

axis: The axis of the parametric and intermediate coordinate systems that are aligned.

angle: The angle from the x' to the x axis measured in a positive sense about the z' axis.

6.54 axisymmetric_volume_2d_element_descriptor

Based on Part 43 [7] 4.4.4, Part 104 [3] 5.4.5, 5.7.13 and 5.7.15.

A **axisymmetric_volume_2d_element_descriptor** is a collection of information that specifies an axisymmetric volume 2D element.

EXPRESS specification:

```
*)
ENTITY axisymmetric_volume_2d_element_descriptor;
       name                    : STRING;
       topology_order          : element_order;
       description             : STRING;
       purpose                 : SET [1:?] OF volume_element_purpose;
       shape                   : element_2d_shape;
END_ENTITY;
(*
```

Attribute definitions:

topology_order: the highest degree polynomial interpolation function used to describe the geomtric shape of any edge of an element, which in turn is used to relate the additional node list of the element to the appropriate toplogy diagram.

description: Additional information about the formulation or purpose of an element.

purpose: The enumerated value specifying the response of a **volume_2d_element_representation.**

shape: The geometric shape of a **volume_3d_element_representation.**

6.55 element_2d_shape

Based on Part 104 [3] 5.3.9. Additional keword *geometric* added to allow for the definition of the element to depend upon its geometry.

If the **element_2d_shape** is **geometric** then the **element_representations** which reference that **element_descriptor** should have a **shape_representation** reference and no **required_node_list.**

EXPRESS specification:

```
*)
TYPE element_2d_shape = ENUMERATION OF
     (quadrilateral, triangle, geometric);
END_TYPE;
(*
```

6.56 plane_volume_2d_element_representation

Based on Part 43 [7] 4.4.12, 4.4.5, Part 104 [3] 5.7.1, 5.7.3 and 5.7.6.

An **plane_volume_2d_element_representation** is a 2D shape in the (i, j) plane of the founding coordinate system with a depth perpendicular to the (i, j) plane. A **plane_volume_2d_element_representation** shall lie in the (i, j) plane of the founding coordinate system.

Other changes have been made as for **volume_3d_element_representation.**

EXPRESS specification:

```
*)
ENTITY plane_volume_2d_element_representation;
       name                       : STRING;
       required_node_list         : OPTIONAL LIST [1:?] OF node;
       model_ref                  : fea_model_2d;
       element_descriptor         : plane_volume_2d_element_descriptor;
       material_properties        : SET [1:?] OF material_property_representation;
       element_properties         : plane_2d_element_property;
       additional_node_list       : OPTIONAL LIST [1:?] OF node;
       material_coordinate_sys    : volume_2d_element_coordinate_system;
       response_property          : OPTIONAL shape_representation;
UNIQUE
  UR1 : model_ref, name;
END_ENTITY;
(*
```

Attribute definitions:

model_ref: An application defined identifier of the **fea_model_2d** which possesses the volume 3d element.

6.57 plane_volume_2d_element_descriptor

Based on Part 43 [7] 4.4.4 , Part 104 [3] 5.4.5, 5.7.13 and 5.7.16.

A **plane_volume_2d_element_descriptor** is a collection of information that specifies a plane volume 2D element.

EXPRESS specification:

```
*)
ENTITY plane_volume_2d_element_descriptor;
       name                    : STRING;
       topology_order          : element_order;
       description             : STRING;
       purpose                 : SET [1:?] OF volume_element_purpose;
       shape                   : element_2d_shape;
       assumption              : plane_2d_element_assumption;
END_ENTITY;
(*
```

Attribute definitions:

topology_order: the highest degree polynomial interpolation function used to describe the geometric shape of any edge of an element, which in turn is used to relate the additional node list of the element to the appropriate topology diagram.

description: Additional information about the formulation or purpose of an element.

purpose: The enumerated value specifying the response of a **plane_volume_2d_element_representation.**

shape: The geometric shape of a **plane_volume_2d_element_representation.**

assumption: The use of a **plane_volume_2d_element_representation** with two dimensional shape to model a volume.

6.58 plane_2d_element_assumption

From Part 104 [3] 5.3.4.

A **plane_2d_element_assumption** is an enumeration of the assumption of the response through the thickness of a finite element.

EXPRESS specification:

```
*)
TYPE plane_2d_element_assumption = ENUMERATION OF
     (plane_stress, plane_strain);
END_TYPE;
(*
```

6.59 volume_2d_element_basis

From Part 104 [3] 5.7.18.

A **volume_element_basis** is the information that form the basis of a volume 2d element.

EXPRESS specification:

```
*)
ENTITY volume_2d_element_basis;
       descriptor               : volume_2d_element_descriptor;
       variable                 : volume_variable;
       cariable_order           : element_order;
       variable_shape_function  : shape_function;
       evaluation_points        : LIST [1:?] OF volume_element_location;
END_ENTITY;
(*
```

Attribute definitions:

descriptor: The association to the information describing a **volume_2d_element_representation.**

variable: The variable to be associated with an order and shape function for a **volume_2d_element_representation.**

variable_order: The mathematical order of the polynomials used to define the variable shape function.

variable_shape_function: The type of polynomial shape function used to interpolate the cariable within a **volume_2d_element_representation.**

evaluation_points: The locations within a**volume_2d_element_representation** where the variable is evaluated using the **variable_shape_function.**

6.60 volume_2d_element_integrated_matrix

Based on Part 104 [3] 5.10.13, 5.4.5 and Part 43 [7] 4.4.4.

A **volume_2d_element_integrated_matrix** is the matrix to be integrated for a volume 2d element and the method of integration.

EXPRESS specification:

```
*)
ENTITY volume_2d_element_integrated_matrix;
       name                     : STRING;
       descriptor               : volume_2d_element_descriptor;
       matrix_property_type     : matrix_property_type;
       integration_description  : STRING;
       integration_definition   : volume_3d_element_field_integration;
END_ENTITY;
(*
```

Attribute definitions:

descriptor: The association to the information describing a **volume_2d_element_representation.**

matrix_property_type: The type of matrix being evaluated.

integration_description: The interpolation rule and integration method.

integration_definition: A definition of the integration within the 3D volume.

6.61 axisymmetric_2d_element_property

Based on Part 43 [7] 4.4.4 and Part 104 [3] 5.13.20.

An **axisymmetric_2d_element_property** is the properties for all types of axisymmetric 2D elements.

EXPRESS specification:

```
*)
ENTITY axisymmetric_2d_element_property;
       name                       : STRING;
       angle                      : REAL;
END_ENTITY;
(*
```

Attribute definitions:

angle: The segment considered in an axisymmetric analysis.

6.62 plane_2d_element_property

Based on Part 43 [7] 4.4.4 and Part 104 [3] 5.13.21.

An **plane_2d_element_property** is the properties for all types of plane 2D elements.

EXPRESS specification:

```
*)
ENTITY plane_2d_element_property;
       name                       : STRING;
       depth                      : REAL;
END_ENTITY;
(*
```

Attribute definitions:

depth: A depth of a plane stress or plane strain section.

6.63 fea_material_representation

Based on Part 104 [3] 5.12.1 but with major changes.

A **fea_material_representation** collects together a set of **material_property_representation** and associates a name with them so they can easily be referenced by multiple elements.

EXPRESS specification:

```
*)
ENTITY fea_material_representation;
       name                       : STRING;
       description                : STRING;
       material_properties        : SET [1:?] OF material_property
                                                    _representation;
END_ENTITY;
(*
```

6.64 material_property_representation

Based on Part 45 [9] 5.3.1 and Part 41 [5] 2.5.3.2. Changed to point directly to a material_property rather than the **SUPERTYPE property_definition.**

A **material_property_representation** associates a **fea_material_property_representation**, i.e. the numerical value or values representing the property, with a **material_property**, which gives the definition of the property being represented, and with a **data_environment** under which this value or values is valid. The attribute **definition** which allows the **material_property_representation** to be associated with a **material_property** has been removed for this first version of the MIDAS data model.

EXPRESS specification:

```
*)
ENTITY material_property_representation;
       used_representation        : fea_material_property_representation;
       dependent_env              : data_environment;
END_ENTITY;
(*
```

Attribute definitions:

definition: The identification of the property that is being represented.

used_representation: the representation of the property which is defined by the **definition** attribute.

dependent_env: the conditions under which a **property_representation** is valid.

6.65 data_environment

Based on Part 45 [9] 5.3.2. Changed to point directly to **fea_material_property_representation** instead of **property_definition_representation.**

A **data_environment** entity is a set of **property_definitions** which allows the conditions which related to one or more properties to be grouped together.

EXPRESS specification:

```
*)
ENTITY data_environment;
       name                        : STRING;
       description                 : STRING;
       elements                    : SET [0:?] OF fea_material_property
                                                  _representation;
END_ENTITY;
(*
```

Attribute definitions:

elements: The set of conditions uder which a **data_environment** is valid.

6.66 fea_material_property_representation

Based on Part 104 [3].

The **fea_material_property_representation** gives the values of a material property.

EXPRESS specification:

```
*)
ENTITY fea_material_property_representation;
       name                        : STRING;
       values                      : fea_constants;
END_ENTITY;
(*
```

Attribute definitions:

values: This attribute gives the actual values of the material property.

6.67 fea_constants

This select type allows the material properties to be either scalar values, second order tensor in three dimensions or fourth order tensor in three dimensions. The definitions of these types are given in FEA definition, Part 104. The *EXPRESS* code only is given in the following sections.

EXPRESS specification:

```
*)
TYPE fea_constants = SELECT
     (scalar, symmetric_tensor2_3d,
      symmetric_tensor4_3d );
END_TYPE;
(*
```

6.68 scalar

Based on Part 104 [3] 7.3.2.

Full explanation given in Part 104.

EXPRESS specification:

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

6.69 symmetric_tensor2_3d

Part 104 [3] 7.3.8./smallskip
Full explanation given in Part 104.

EXPRESS specification:

```
*)
TYPE symmetric_tensor2_3d = SELECT
     (isotropic_symmetric_tensor2_3d,
      orthotropic_symmetric_tensor2_3d,
      anisotropic_symmetric_tensor2_3d);
END_TYPE;
(*
```

6.70 isotropic_symmetric_tensor2_3d

Based on Part 104 [3] 7.3.9.

Full explanation given in Part 104.

EXPRESS specification:

```
*)
TYPE isotropic_symmetric_tensor2_3d = REAL;
END_TYPE;
(*
```

6.71 orthotropic_symmetric_tensor2_3d

Based on Part 104 [3] 7.3.10.

Full explanation given in Part 104.

EXPRESS specification:

```
*)
TYPE orthotropic_symmetric_tensor2_3d = ARRAY [1:3] OF REAL;
END_TYPE;
(*
```

6.72 anisotropic_symmetric_tensor2_3d

Based on Part 104 [3] 7.3.11.

Full explanation given in Part 104.

EXPRESS specification:

```
*)
TYPE anisotropic_symmetric_tensor2_3d = ARRAY [1:6] OF REAL;
END_TYPE;
(*
```

6.73 symmetric_tensor4_3d

Based on Part 104 [3] 7.4.1. Some of the select types have been removed for this first version of the MIDAS data model. Extra types may be added in at a later date.

Full explanation given in Part 104.

EXPRESS specification:

```
*)
TYPE symmetric_tensor4_3d = SELECT
     (anisotropic_symmetric_tensor4_3d,
      fea_isotropic_symmetric_tensor4_3d);
END_TYPE;
(*
```

6.74 anisotropic_symmetric_tensor4_3d

Based on Part 104 [3] 7.4.2.

Full explanation given in Part 104.

EXPRESS specification:

```
*)
TYPE anisotropic_symmetric_tensor4_3d = ARRAY [1:21] OF REAL;
END_TYPE;
(*
```

6.75 fea_isotropic_symmetric_tensor4_3d

Based on Part 104 [3] 7.4.3.

Full explanation given in Part 104.

EXPRESS specification:

```
*)
TYPE fea_isotropic_symmetric_tensor4_3d = ARRAY [1:2] OF REAL;
END_TYPE;
(*
```

6.76 element_group

Based on Part 104 [3] 5.14.2 and 5.14.1.

An **element_group** is a group containing only elements.

EXPRESS specification:

```
*)
ENTITY element_group;
       group_id                : STRING;
       description             : STRING;
       elements                : SET [1:?] OF element_representation;
END_ENTITY;
(*
```

Attribute definitions:

group_id: An application defined identifier for the group, and is unique within the FEA information model.

description: The group. For example, possible values of this attribute might be red, set 1, or wing. It is whatever the analysist chooses as a label for group identification.

elements: The set of elements in the group.

6.77 node_group

Based on Part 104 [3] 5.14.3 and 5.14.1.

An **node_group** is a group containing only nodes.

EXPRESS specification:

```
*)
ENTITY node_group;
       group_id                : STRING;
       description             : STRING;
       nodes                  : SET [1:?] OF node;
END_ENTITY;
(*
```

Attribute definitions:

group_id: An application defined identifier for the group, and is unique within the FEA information model.

description: The group. For example, possible values of this attribute might be red, set 1, or wing. It is whatever the analysist chooses as a label for group identification.

nodes: The set of nodes in the group.

6.78 fea_group

Based on Part 104 [3] 5.14.1.

Created to because of removal of supertype structure between **element_group** and **node_group**, to allow entities to refer to either type.

EXPRESS specification:

```
*)
TYPE fea_group = SELECT
      (element_group, node_group);
END_TYPE;
(*
```

7 Data Model for Finite Element Controls

7.1 volume_3d_face

From Part 104 [3] 6.3.3.

A **volume_3d_face** is an element face of a volume 3D element.

EXPRESS specification:

```
*)
TYPE volume_3d_face = INTEGER;
WHERE
   WR1: volume_3d_face >= 1 and volume_3d_face <=6;
END_TYPE;
(*
```

7.2 field_value

From Part 104 [3] 6.3.9.

A **field_value** is the value of the field variable.

EXPRESS specification:

```
*)
TYPE field_value = SELECT
      (unspecified_value,
       scalar,
       tensor1_2d,
       tensor1_3d,
       anisotropic_symmetric_tensor2_2d,
       symmetric_tensor2_3d);
END_TYPE;

(*
```

7.3 unspecified_value

From Part 104 [3] 6.3.10.

EXPRESS specification:

```
*)
TYPE unspecified_value = ENUMERATION OF
      (unspecified);
END_TYPE;

(*
```

7.4 tensor1_2d

Based on Part 104 [3] 7.3.4.

Full explanation given in Part 104.

EXPRESS specification:

```
*)
TYPE tensor1_2d = ARRAY [1:2] OF REAL;
END_TYPE;
(*
```

7.5 tensor1_3d

Based on Part 104 [3] 7.3.5.

Full explanation given in Part 104.

EXPRESS specification:

```
*)
TYPE tensor1_3d = ARRAY [1:3] OF REAL;
END_TYPE;
(*
```

7.6 anisotropic_symmetric_tensor2_2d

Based on Part 104 [3] 7.3.7.

Full explanation given in Part 104.

EXPRESS specification:

```
*)
TYPE anisotropic_symmetric_tensor2_2d = ARRAY [1:3] OF REAL;
END_TYPE;
(*
```

7.7 volume_3d_element_representation_or_descriptor

Part 104 [3] 6.3.49.

A **volume_3d_element_representation_or_descriptor** is either an element representation or an element descriptor for output request selection for volume 3D elements. If an element descriptor is referenced then all elements that reference that descriptor will have the same output request.

EXPRESS specification:

```
*)
TYPE volume_3d_element_representation_or_descriptor = SELECT
      (volume_3d_element_representation,
       volume_3d_element_descriptor);
END_TYPE;

(*
```

7.8 control

Based on Part 104 [3] 6.4.1.

A **control** is administrative information for a model, and associates analysis controls and results with a unique model.

EXPRESS specification:

```
*)
ENTITY control;
       model_ref                : fea_model;
       control_id               : STRING;
       creating_software        : STRING;
       description              : STRING;
       user_defined_control    : SET [1:?] OF STRING;
END_ENTITY;
(*
```

7.9 control_analysis_step

Based on Part 104 [3] 6.4.3, 6.4.2, 6.4.7 and 6.4.15.

A **control_analysis_step** is a single step in an analysis.

EXPRESS specification:

```
*)
ENTITY control_analysis_step;
       analysis_control         : control;
       step_id                  : STRING;
       sequence                 : integer;
       initial_state            : state;
       description              : STRING;
       final_input_state       : state;
UNIQUE
  UR1: analysis_control, sequence;
  UR2: analysis_control, step_id;
END_ENTITY;
(*
```

Attribute definitions:

sequence: A sequence number that determines the order of processing for the control analysis step.

initial_state: The state of the model at the beginning of the step. This includes information such as reference temperatures and initial strains or stresses. For an initial equilibrium state of zero stress or strain it is only necessary to identify the initial state. The state will then have no referencing state definitions.

final_input_state: The final equilibrium state for the static load increment. The final state includes all field and node stated definitions applied to the model. This state is a specified state without any analysis output information until an analysis has been carried out.

7.10 output_request_set

From Part 104 [3] 6.4.16.

A **output_request_set** is the computer readable output to be produced by a finite element analysi system as follows:

- the attributes of the entity provide administrative information for the set of output;
- details of the information to be produced by the finite element analysis system are supplied by the **state_definition_without_value** entities which reference it.

EXPRESS specification:

```
*)
ENTITY output_request_set;
       steps                  : SET [1:?] OF control_analysis_step;
       output_set_id          : STRING;
       description            : STRING;
END_ENTITY;
(*
```

Attribute definitions:

steps: The analysis control steps to which the output requests apply.

output_set_id: The identifier for the output set.

description: Additional information about the output set.

7.11 single_point_constraint_element

Based on Part 104 [3] 6.4.10 and 6.4.9.

A **single_point_constraint_element** is a single point constraint which sets values for one or more degrees of freedom at a single node.

EXPRESS specification:

```
*)
ENTITY single_point_constraint_element;
       control                : control;
       element_id             : STRING;
       steps                  : SET [1:?] OF control_analysis_step;
       required_node          : node;
       coordinate_system      : OPTIONAL fea_axis2_placement_3d;
       freedoms_and_value     : SET [1:?] OF freedom_and_coefficient;
       description            : STRING;
END_ENTITY;
(*
```

Attribute definitions:

control: The control of which the constraint element is a part.

element_id: The identifier for hte constraint element.

steps: The analysis control steps to which the constraint element applies.

required_node: The node which is being constrained.

coordinate_system: The coordinate system with respect to which the constraint freedoms are defined.

freedoms_and_values: The freedom and value pair imposed by the constraint.

description: Additional information about the single point constraint element.

7.12 freedom_and_coefficient

Based on Part 104 [3] 6.4.13.

A **freedom_and_coefficient** is the degree of freedom and the corresponding coefficient value of a constraint equation.

EXPRESS specification:

```
*)
ENTITY freedom_and_coefficient;
       freedom                    : degree_of_freedom;
       a                          : OPTIONAL REAL;
END_ENTITY;
(*
```

Attribute definitions:

freedom: The degree of freedom that the value is associated with.

a: The nodal freedom coefficient *a* associated with the degree of freedom.

7.13 state

Based on Part 104 [3] 6.6.1. It replaces the SUPERTYPE construct.

A **state** is the state of the model as follows:

- the attributes of the entity provide administrative information for the state;
- details of the state are supplied by the **state_definition** entities which reference it.

Some of the select types (originally subtypes) have been removed for this version of the MIDAS data model.

EXPRESS specification:

```
*)
TYPE state = SELECT
     (specified_state, calculated_state);
END_TYPE;
(*
```

7.14 specified_state

Based on Part 104 [3] 6.6.3 and 6.6.1.

A **specified_state** is all of the information for the state that is specified, and none of it is the result of a previous calculation represented within the information model.

EXPRESS specification:

```
*)
ENTITY specified_state;
       state_id                  : STRING;
       description               : STRING;
END_ENTITY;
(*
```

7.15 calculated_state

Based on Part 104 [3] 6.6.4 and 6.6.1.

A **calculated_state** is some of the information for the state that has been calculated by a previous analys step. It forms part of analysis results and may form part of the loading for a following step.

EXPRESS specification:

```
*)
ENTITY calculated_state;
       state_id                  : STRING;
       description               : STRING;
END_ENTITY;
(*
```

7.16 volume_3d_element_constant_specified_variable_value

Based on Part 104 [3] 6.7.10, 6.7.6, 6.7.5, 6.7.4, 6.7.3, 6.7.2 and 6.7.1.

A **volume_3d_element_constant_specified_variable_value** is a state of the model that specifies the value of a variable which is constant over a volume 3D element.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_constant_specified_variable_value;
       state_with_value          : OPTIONAL state;
       output_set                : OPTIONAL output_request_set;
       element                   : volume_3d_element_representation
                                                 _or_descriptor;
       simple_value              : field_value;
       variable                  : volume_variable;
END_ENTITY;
(*
```

Attribute definitions:

state_with_value: The model state containing calculated values. This must be present if and only if this entity is a state definition with value, i.e. a loading.

output_set: the output set which the state definition defines. This must be present if and only if this entity is a state definition without value, i.e. a request for calculation by the analysis.

element: The element for which the values are specified, or the element for which values are to be calculated.

simple_value: The value of the field variable. This only exists when the entity is a state definition with value.

variable: The type of field variable being specified.

7.17 volume_3d_element_nodal_specified_variable_value

Based on Part 104 [3] 6.7.11, 6.7.6, 6.7.5, 6.7.4, 6.7.3, 6.7.2 and 6.7.1.

A **volume_3d_element_nodal_specified_variable_value** is a state of the model that specifies the value of a variable for the nodes of a volume 3D element. The variable is interpolated within the element using the appropriate shape functions specified for the variable.

The values are supplied for each of the nodes in the attribute **required_node_list** of entity **element**. Further values are supplied for each node in attribute **additional_node_list** which is not a dummy.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_nodal_specified_variable_value;
       state_with_value         : OPTIONAL state;
       output_set               : OPTIONAL output_request_set;
       element                  : volume_3d_element_representation
                                          _or_descriptor;
       values                   : LIST [1:?] OF field_value;
       additional_node_values   : BOOLEAN;
       variable                 : volume_variable;
END_ENTITY;
(*
```

Attribute definitions:

state_with_value: The model state containing calculated values.This must be present if and only if this entity is a state definition with value, i.e. a loading.

output_set: the output set which the state definition defines. This must be present if and only if this entity is a state definition without value, i.e. a request for calculation by the analysis.

element: The element for which the values are specified, or the element for which values are to be calculated.

values: The values of the field variable. This only exists when the entity is a state definition with value.

additional_node_values: Indicates whether values are given for all nodes of the element (TRUE) or for just the required nodes (FALSE).

variable: The type of field variable being specified.

7.18 volume_3d_element_boundary_constant_specified_variable_value

Based on Part 104 [3] 6.7.14, 6.7.6, 6.7.5, 6.7.4, 6.7.3, 6.7.2 and 6.7.1.

A **volume_3d_element_boundary_constant_specified_variable_value** is a state of the model that specifies the value of a variable which is constant over the whole of a volume 3D element face.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_boundary_constant_specified_variable_value;
       state_with_value          : OPTIONAL state;
       output_set                : OPTIONAL output_request_set;
       element                   : volume_3d_element_representation
                                                     _or_descriptor;
       simple_value              : field_value;
       variable                  : boundary_variable;
       element_face              : volume_3d_face;
END_ENTITY;
(*
```

Attribute definitions:

state_with_value: The model state containing calculated values.This must be present if and only if this entity is a state definition with value, i.e. a loading.

output_set: the output set which the state definition defines. This must be present if and only if this entity is a state definition without value, i.e. a request for calculation by the analysis.

element: The element for which the values are specified, or the element for which values are to be calculated.

value: The value of the field variable. This only exists when the entity is a state definition with value.

variable: The type of field variable being specified.

element_face: The element face for which the values are specified.

7.19 volume_3d_element_boundary_nodal_specified_variable_value

Based on Part 104 [3] 6.7.15, 6.7.6, 6.7.5, 6.7.4, 6.7.3, 6.7.2 and 6.7.1.

A **volume_3d_element_boundary_nodal_specified_variable_value** is a state of the model that specifies the value of a variable for the nodes on an element face of a volume 3D element. The variable is interpolated within the element using the appropriate shape functions specified for the variable. Values are supplied for the nodes on the element face, in the order established graphically in Appendix B.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_boundary_nodal_specified_variable_value;
       state_with_value          : OPTIONAL state;
       output_set                : OPTIONAL output_request_set;
       element                   : volume_3d_element_representation
```

```
                                                  _or_descriptor;
        values                      : LIST [1:?] OF field_value;
        additional_node_values      : BOOLEAN;
        variable                    : boundary_variable;
        element_face                : volume_3d_face;
END_ENTITY;
(*
```

Attribute definitions:

state_with_value: The model state containing calculated values. This must be present if and only if this entity is a state definition with value, i.e. a loading.

output_set: the output set which the state definition defines. This must be present if and only if this entity is a state definition without value, i.e. a request for calculation by the analysis.

element: The element for which the values are specified, or the element for which values are to be calculated.

values: The values of the field variable. This only exists when the entity is a state definition with value.

additional_node_values: Indicates whether values are given for all nodes of the element (TRUE) or for just the required nodes (FALSE).

variable: The type of field variable being specified.

element_face: The element face for which the values are specified.

7.20 boundary_variable

Based on Part 104 [3] 6.3.12.

A **boundary_variable** is a field variable on a boundary of an element.

EXPRESS specification:

```
*)
TYPE boundary_variable = SELECT
      (boundary_surface_scalar_variable,
       boundary_surface_vector_3d_variable);
END_TYPE;
(*
```

7.21 boundary_surface_scalar_variable

Based on Part 104 [3] 6.3.23.

A **boundary_surface_scalar_variable** is a scalar field variable that is evaluated at a point on the element face of a surface or volume element.

EXPRESS specification:

```
*)
TYPE boundary_surface_scalar_variable = ENUMERATION OF
     (pressure);
END_TYPE;
(*
```

7.22 boundary_surface_vector_3d_variable

Based on Part 104 [3] 6.3.35.

A **boundary_surface_vector_3d_variable** is a 3D vector field variable that is evaluated at a point on the element face of a volume element.

EXPRESS specification:

```
*)
TYPE boundary_surface_vector_3d_variable = ENUMERATION OF
     (applied_force_per_unit_area,
      applied_moment_per_unit_area);
END_TYPE;
(*
```

7.23 volume_3d_node_field_variable_definition

Based on Part 104 [3] 6.7.86, 6.7.85, 6.7.4, 6.7.3, 6.7.2 and 6.7.1.

A **volume_3d_node_field_variable_definition** is a state of the model that specifies the values of a field variable at the nodes of a volume 3d element to represent nodal averaged values.

EXPRESS specification:

```
*)
ENTITY volume_3d_node_field_variable_definition;
      state_with_value          : OPTIONAL state;
      output_set                : OPTIONAL output_request_set;
      node                      : node;
      group                     : OPTIONAL element_group;
      simple_value              : field_value;
      variable                  : volume_variable;
      coordinate_system         : OPTIONAL volume_3d_element_coordinate_system;
END_ENTITY;
(*
```

Attribute definitions:

state_with_value: The model state containing calculated values. This must be present if and only if this entity is a state definition with value, i.e. a loading.

output_set: the output set which the state definition defines. This must be present if and only if this entity is a state definition without value, i.e. a request for calculation by the analysis.

node: The node for which the value is specified,or the node for which the value is to be calculated.

group: The elements connected to the node for which the field variable value is valid. If this attribute is omitted then the variable is valid for all elements connected to the node.

simple_value: The value of the field variable. This only exists when the entity is a state definition with value.

variable: The type of field variable being specified.

coordinate_system: The coordinate system for the value. This shall be specified if any of the specified values are not scalar.

7.24 nodal_freedom_and_value_definition

Based on Part 104 [3] 6.7.100, 6.7.2 and 6.7.1.

A **nodal_freedom_and_value_definition** is a state of the model that specifies information at a node of the model with respect to the solution degrees of freedom that are not part of a field discretization. The information may be:

- values for the solution degrees of freedom, such as x translation or z rotation,
- loads applied in the direction of the solution degrees of freedom such as values for x force or z moment. These may be loads applied to a node by elements of the model or loads applied to a node from outside the model.

This entity should not be used to represent a displacement field that has been discretized at the model nodes. The **field_variable_node_definition** should be used for that purpose.

EXPRESS specification:

```
*)
ENTITY nodal_freedom_and_value_definition;
       state_with_value         : OPTIONAL state;
       output_set               : OPTIONAL output_request_set;
       node                     : node;
       coordinate_system        : OPTIONAL fea_axis2_placement_3d;
       freedoms_and_values      : SET [1:?] OF freedom_and_value;
END_ENTITY;
(*
```

Attribute definitions:

state_with_value: The model state containing calculated values. This must be present if and only if this entity is a state definition with value, i.e. a loading.

output_set: the output set which the state definition defines. This must be present if and only if this entity is a state definition without value, i.e. a request for calculation by the analysis.

node: the node for which the values are specified, or the node for which the values are to be calculated.

coordinate_system: The coordinate system for the value. This shall be specified if any of the specified values are not scalar.

freedoms_and_values: The degree of freedom and the matching value.

7.25 freedom_and_value

Based on Part 104 [3] 6.7.105.

A **freedom_and_value** is the degree of freedom and the corresponding value of element nodal action at a node of an element. The type of action is indicated by the degrees of freedom. Hence for freedom *x_displacement* the x force value is specified.

EXPRESS specification:

```
*)
ENTITY freedom_and_value;
       freedom                  : degree_of_freedom;
       simple_value             : OPTIONAL REAL;
END_ENTITY;
(*
```

8 Data Model for Finite Element Analysis Results

Analysis results for a finite element analysis consist of the response of a model to a control as calculated by a finite element analysis application. All model states calculated by the analysis application shall be linked to a result.

8.1 result

From Part 104 [3] 6.5.1.

A **result** is the administrative information for analysis results.

EXPRESS specification:

```
*)
ENTITY result;
        result_id               : STRING;
        creating_software       : STRING;
        description             : STRING;
END_ENTITY;
(*
```

8.2 result_analysis_step

Based on Part 104 [3] 6.5.2 and 6.5.3.

A **result_analysis_step** is the results from an analysis sub-step. It is the final state of the step.

EXPRESS specification:

```
*)
ENTITY result_analysis_step;
        analysis_result         : result;
        state                   : calculated_state;
END_ENTITY;
(*
```

```
*)
END_SCHEMA;
(*
```

References

[1] "Data Modelling for Electromagnetic and Stress Analysis Integration", Mrs D Thomas and Dr C Greenough, March 1996, Rutherford Appleton Laboratory Technical Report (*To be published*).

[2] "Common Data Model for MIDAS, Version 1.1", Mrs D Thomas, Dr C Greenough and J van Maanen, *MIDAS Project Report*, MIDAS.RAL.94.5,

[3] ISO TC184/SC4/WG3 N263x (P9), STEP Part 104, Integrated Application Resources: Finite Element Analysis, 9 September 1994.

[4] ISO TC184/SC4/WG3 N350 (P9), Part 209, Composite and Metallic Structural Analysis and Related Design, 17 October 1994.

[5] ISO 10303, Part41, "Integrated generic resources: Fundamentals of product description and support", IS Draft, 23 August 1994.

[6] ISO 10303, Part42, "Integrated generic resources: Geometric and topological representation", IS Draft, 29 August 1994.

[7] ISO 10303, Part43, "Integrated generic resources: Representation Structures", IS Draft, 15 August 1994.

[8] ISO 10303, Part44, "Integrated generic resources: Product Structure Configuration", IS Draft, August 1994.

[9] ISO/TC184/SC4/WG3 N258 , "Integrated generic resources: Materials",

A STEP Data Model Entities Used

This appendix lists all the STEP data model entities which have been used in constructing the MIDAS data model. A separate list is given for each of the referenced documents. For each document the table gives the section number of the entity, the entity name and a code which designates the level of change made to that the entity for inclusion in the MIDAS data model. The meaning of this code is as follows:

- U the entity is unchanged from the STEP entity.
- M minor changes have been made according to the general principles described in the main report.
- C specific changes have made to the entity which will vary in type and degree for each entity changed, these include extensions to entities to deal with the additional requirements for electromagnetic analysis.

A.1 Part 104 Finite Element Analysis

Section no.	Entity or Type name	Changes
5.3.1	axi_or_plane	U
5.3.2	coordinate_system_type	U
5.3.3	element_order	U
5.3.4	plane_2d_elememt_assumption	U
5.3.5	volume_element_purpose	C
5.3.8	volume_3d_element_shape	C
5.3.9	element_2d_shape	C
5.3.10	matrix_property_type	U
5.3.14	degree_of_freedom	C
5.3.16	integration_rule	U
5.3.17	shape_function	U
5.3.18	additional_node	M
5.3.20	volume_2d_element_representation	U
5.3.23	volume_2d_element_descriptor	U
5.3.26	volume_3d_element_coordinate_system	U
5.3.27	volume_2d_element_coordinate_system	U
5.4.1	fea_model	C
5.4.2	fea_3d_model	C
5.4.3	fea_2d_model	C
5.4.4	structural_response_property	M
5.4.5	fea_representation_item	M
5.5.2	fea_axis2_placement_3d	M
5.6.1	node_representation	M
5.6.2	node_shape_representation	M
5.6.3	node	C

5.7.1	element_representation	C
5.7.2	element_shape_representation	C
5.7.3	higher_order_element_representation	C
5.7.4	volume_3d_element_representation	C
5.7.5	axisymmetric_volume_2d_element_representation	C
5.7.6	plane_volume_3d_element_representation	C
5.7.13	element_descriptor	M
5.7.14	volume_3d_element_descriptor	M
5.7.15	axisymmetric_volume_2d_element_descriptor	M
5.7.16	plane_volume_2d_element_descriptor	M
5.7.17	volume_3d_element_basis	U
5.7.18	volume_2d_element_basis	U
5.9.2	arbitrary_volume_3d_element_coordinate_system	M
5.9.3	parametric_volume_3d_element_coordinate_system	M
5.9.4	arbitrary_volume_2d_element_coordinate_system	M
5.9.5	parametric_volume_2d_element_coordinate_system	M
5.9.18	euler_angles	M
5.10.6	volume_3d_element_integrated_matrix	M
5.10.8	volume_3d_element_field_integration	U
5.10.9	element_integration_algebraic	U
5.10.10	volume_3d_element_field_integration_rule	U
5.10.11	volume_3d_element_field_integration_explicit	U
5.10.12	volume_position_weight	U
5.10.13	volume_2d_element_integrated_matrix	M
5.11.1	fea_parametric_point	M
5.11.2	volume_element_location	U
5.12.1	fea_material_representation	C
5.13.20	axisymmetric_2d_element_property	M
5.13.21	plane_2d_element_property	M
5.14.1	fea_group	M
5.14.2	element_group	M
5.14.3	node_group	M
6.3.3	volume_3d_face	U
6.3.9	field_value	U
6.3.10	unspecified_value	U
6.3.12	boundary_variable	M
6.3.14	volume_variable	C
6.3.21	volume_scalar_variable	C
6.3.23	boundary_surface_scalar_variable	M
6.3.25	volume_angular_variable	C
6.3.33	volume_vector_3d_variable	C
6.3.35	boundary_surface_vector_3d_variable	M

6.3.40	volume_tensor2_3d_variable	U
6.3.49	volume_3d_element_representation_or_descriptor	U
6.4.1	control	M
6.4.2	analysis_step	M
6.4.3	control_analysis_step	M
6.4.7	control_linear_static_analysis_step	M
6.4.9	constraint_element	M
6.4.10	single_point_constraint_element	M
6.4.13	freedom_and_coefficient	M
6.4.15	control_static_load_increment_process	C
6.4.16	output_request_set	C
6.5.1	result	U
6.5.2	result_analysis_step	M
6.5.3	result_linear_static__analysis_step	M
6.6.1	state	M
6.6.3	specified_state	M
6.6.3	calculated_state	M
6.7.1	state_definition	M
6.7.2	state_definition_without_value	M
6.7.3	state_definition_with_value	M
6.7.4	field_variable_definition	M
6.7.5	field_variable_element_definition	M
6.7.6	volume_3d_element_field_variable_definition	M
6.7.10	volume_3d_element_constant_specified_variable_value	M
6.7.11	volume_3d_element_nodal_specified_variable_value	M
6.7.14	volume_3d_element_boundary_constant_specified_variable_value	M
6.7.15	volume_3d_element_boundary_nodal_specified_variable_value	M
6.7.85	field_variable_node_definition	M
6.7.86	volume_3d_node_field_variable_definition	M
6.7.100	nodal_freedom_and_value_definition	M
6.7.105	freedom_and_value	M
7.3.2	scalar	M
7.3.4	tensor1_2d	M
7.3.5	tensor1_3d	M
7.3.7	anisotropic_symmetric_tensor2_2d	M
7.3.8	symmetric_tensor2_3d	U
7.3.9	isotropic_symmetric_tensor2_3d	M
7.3.10	orthotropic_symmetric_tensor2_3d	M
7.3.11	anisotropic_symmetric_tensor2_3d	M
7.4.1	symmetric_tensor4_3d	C
7.4.2	anisotropic_symmetric_tensor4_3d	M
7.4.3	fea_isotropic_symmetric_tensor4_3d	M

A.2 Part 41 Fundamentals of product description and support

Section no.	Entity or Type name	Changes
2.4.3.1	characterized_definition	M
2.4.3.3	shape_definition	M
2.5.3.1	shape_representation	C
2.5.3.2	property_definition_representation	M
4.13.3.1	identifier	M
4.13.3.2	label	M
4.13.3.3	text	M
4.14.3.1	measure_value	M
4.14.3.22	unit	U
4.14.3.23	si_unit_name	U
4.14.3.24	si_prefix	U
4.14.4.1	named_unit	C
4.14.4.2	si_unit	M
4.14.4.3	conversion_based_unit	C
4.14.4.17	dimensional_exponents	U
4.14.4.18	derived_unit_element	U
4.14.4.19	derived_unit	U
4.14.4.20	global_unit_assigned_context	M
4.14.4.21	measure_with_unit	C

A.3 Part 42 Geometric and topological representation

Section no.	Entity name	Changes
4.4.1 hline 4.4.2	geometric_representation_context geometric_representation_item	M C
4.4.3	point	M
4.4.4	cartesian_point	M
4.4.10	direction	M
4.4.12	placement	M
4.4.14	axis2_placement_2d	M
4.4.15	axis2_placement_3d	M
5.4.1	topological_representation_item	C
5.4.2	vertex	M
5.4.3	vertex_point	M
5.4.4	edge	M
5.4.7	path	M
5.4.10	loop	M
5.4.12	edge_loop	M
5.4.13	poly_loop	M
5.4.14	face_bound	M

5.4.16	face	M
5.4.20	connected_face_set	M
5.4.25	closed_shell	M
5.4.26	oriented_closed_shell	M
6.4.1	solid_model	M
6.4.2	manifold_solid_rep	C
6.4.3	brep_with_voids	C
6.4.4	faceted_brep	C

A.4 Part 43 Representation Structures

Section no.	Entity name	Changes
4.4.2	representation_context	M
4.4.4	representation_item	C
4.4.5	representation	M
4.4.12	definitional_representation	M

A.5 Part 45 Materials

Section no.	Entity name	Changes
4.4.1	material_property	C
5.3.1	material_property_representation	C
5.3.2	data_environment	C