ICS 35.240.50
L 67

中华人民共和国国家标准

GB/T 16656.502—2005/ISO 10303-502:2000

工业自动化系统与集成 产品数据表达与交换 第502部分：应用解释构造：基于壳的线框

Industrial automation systems and integration—Product data representation and exchange—Part 502:Application interpreted construct:Shell-based wireframe

(ISO 10303-502:2000,IDT)

2005-09-12 发布　　2006-04-01 实施

中华人民共和国国家质量监督检验检疫总局
中国国家标准化管理委员会　发布

中华人民共和国
国家标准
工业自动化系统与集成
产品数据表达与交换 第502部分：
应用解释构造：基于壳的线框
GB/T 16656.502—2005/ISO 10303-502:2000
*
中国标准出版社出版发行
北京复兴门外三里河北街16号
邮政编码：100045
网址 www.bzcbs.com
电话：68523946 68517548
中国标准出版社秦皇岛印刷厂印刷
各地新华书店经销
*
开本 880×1230 1/16 印张 2 字数 47千字
2006年4月第一版 2006年4月第一次印刷
*
书号：155066·1-27206 定价 16.00 元

GB/T 16656.502—2005/ISO 10303-502:2000

GB/T 16656.502-2005

前　言

GB/T 16656《工业自动化系统与集成　产品数据表达与交换》现已批准和发布的有以下26个部分：

——第1部分：概述与基本原理；
——第11部分：描述方法：EXPRESS语言参考手册；
——第21部分：实现方法：交换结构的纯正文编码；
——第31部分：一致性测试方法论与框架：基本概念；
——第32部分：一致性测试方法论与框架：对测试实验室与客户的要求；
——第34部分：一致性测试方法论与框架：应用协议实现的抽象测试方法；
——第41部分：集成通用资源：产品描述与支持原理；
——第42部分：集成通用资源：几何与拓扑表达；
——第43部分：集成通用资源：表达结构；
——第44部分：集成通用资源：产品结构配置；
——第45部分：集成通用资源：材料；
——第46部分：集成通用资源：可视化显示；
——第47部分：集成通用资源：形状变化公差；
——第49部分：集成通用资源：工艺过程结构和特性；
——第101部分：集成应用资源：绘图；
——第105部分：集成应用资源：运动学；
——第201部分：应用协议：显式绘图；
——第202部分：应用协议：相关绘图；
——第203部分：应用协议：配置控制设计；
——第501部分：应用解释构造：基于边的线框
——第502部分：应用解释构造：基于壳的线框
——第503部分：应用解释构造：几何有界的二维线框；
——第513部分：应用解释构造：基本边界表达；
——第520部分：应用解释构造：相关绘图元素。
——第1001部分：应用模块：外观赋值
——第1006部分：应用模块：基础表达

本部分是GB/T 16656的第502部分。

GB/T 16656对应ISO 10303。GB/T 16656各部分的编号与ISO 10303各部分的编号相同。ISO 10303是一个庞大的标准，目前包括121个部分，其目录见附录NA。GB/T 16656的本部分等同采用国际标准ISO 10303-502:2000《工业自动化系统与集成　产品数据表达与交换　第502部分　应用解释构造　基于壳的线框》，其技术内容和结构与ISO 10303-502:2000保持一致。只是为了让标准使用者了解ISO 10303的总体结构，本部分将ISO网站上给出的ISO 10303各部分的目录编入到了附录NA。

本部分的附录A、附录B为规范性附录。

本部分的附录C、附录D、附录NA为资料性附录。

本部分由中国机械工业联合会提出。

本部分由全国工业自动化系统与集成标准化技术委员会归口。

本部分主要起草单位:中国标准化研究院。

本部分主要起草人:董连续、洪岩。

引　言

GB/T 16656 是一项计算机可解释的产品数据表达与交换标准，其目的是提供一种中性机制，能够在产品全生命周期内描述产品，并独立于任何特定系统。这种描述的本质使得它不仅适合中性文件的交换，而且适合作为实现和共享产品数据库及文件存档的基础。

本部分是应用解释构造系列的组成部分。为了产品数据能在多个应用相关环境中使用，应用解释构造(AIC)给出了支持特定功能的解释构造逻辑组合。解释构造是对在应用协议中支持共享信息需求的集成资源的通用解释。

本部分使用一组壳为边界的三维线框模型对产品形状的描述规定了应用解释构造。

目　次

工业自动化系统与集成 产品数据表达与交换 第502部分：应用解释构造：基于壳的线框

1 范围

GB/T 16656的本部分规定了对集成资源的解释，以满足使用三维线框模型表达产品形状，该三维线框模型的边界由一组壳所界定。

本部分适用于：

——由边和顶点所描述的线框模型表达，这些边仅仅在其顶点相交。

——由一个或多个壳组成的线框模型表达，这些壳只能在顶点或边界处重叠或相交。

——在三维坐标空间所定义的点。

——在三维坐标空间所定义的包括B样条在内的曲线。

——单一线框模型表达或线框模型组合表达。

本部分不适用于：

——在二维坐标空间所定义的几何；

——曲面几何；

——实体几何；

——对所删除的构造的引用。

2 规范性引用文件

下列文件中的条款通过GB/T 16656的本部分的引用而成为本部分的条款。凡是注日期的引用文件，其随后所有的修改单(不包括勘误的内容)或修订版均不适用于本部分，然而，鼓励根据本部分达成协议的各方研究是否可使用这些文件的最新版本。凡是不注日期的引用文件，其最新版本适用于本部分。

GB/T 16262—1996 信息处理系统 开放系统互连 抽象语法记法一(ASN.1)规范(idt ISO 8824:1990)

GB/T 16656.1—1998 工业自动化系统与集成 产品数据表达与交换 第1部分：概述与基本原理(idt ISO 10303:1994)

GB/T 16656.11—1996 工业自动化系统与集成 产品数据表达和交换 第11部分：描述方法：EXPRESS语言参考手册(eqv ISO/DIS 10303-11:1993)

GB/T 16656.41—1999 工业自动化系统与集成 产品数据表达和交换 第41部分：集成通用资源：产品描述与支持原理(idt ISO 10303-41:1994)

GB/T 16656.42—1998 工业自动化系统和集成 产品数据表达与交换 第42部分：集成通用资源：几何与拓扑表达(idt ISO 10303-42:1994)

GB/T 16656.43—1999 工业自动化系统与集成 产品数据表达和交换 第43部分：集成通用资源：表达结构(idt ISO 10303-43:1994)

GB/T 16656.202—2000 工业自动化系统与集成 产品数据表达与交换 第202部分：应用协议：相关绘图(eqv ISO 10303-202:1996)

3 术语、定义和缩略语

3.1 GB/T 16656.1 所定义的术语

GB/T 16656.1 所定义的以下术语适用于本部分：

——应用 application；
——应用相关环境 application context；
——应用协议 application protocol；
——实现方法 implementation method；
——集成资源 integrated resource；
——解释 interpretation；
——模型 model；
——产品 product；
——产品数据 product data；

3.2 GB/T 16656.202 所定义的术语

GB/T 16656.202 所定义的以下术语适用于本部分：

——应用解释构造(AIC)。

3.3 缩略语

以下缩略语适用于 GB/T 16656 本部分：

AIC 应用解释构造 application interpreted construct
AP 应用协议 application protocol

4 EXPRESS 短表

本章规定了 EXPRESS 模式，所采用的元素取自集成资源并构成本部分的类型、实体描述和函数。

注 1：可以包括本 AIC 未引入的集成资源所规定的子类和项。通过使用 GB/T 16656.11 隐式接口规则，从子类树或选表中删除构造。

EXPRESS 描述：

```
*)
SCHEMA aic_shell_based_wireframe;

USE FROM geometric_model_schema          -- GB/T 16656.42
  (shell_based_wireframe_model);

USE FROM geometry_schema                 --GB/T 16656.42
  (axis2_placement_3d,
   b_spline_curve_with_knots,
   bezier_curve,
   cartesian_transformation_operator_3d,
   circle,
   conic,
   curve,
   curve_replica,
   ellipse,
```

```
    geometric_representation_context,
    hyperbola,
    line,
    offset_curve_3d,
    parabola,
    point,
    point_replica,
    polyline,
    quasi_uniform_curve,
    rational_b_spline_curve,
    uniform_curve);

USE FROM product_property_representation_schema      —GB/T 16656.41
  (shape_representation);

USE FROM representation_schema      —GB/T 16656.43
  (mapped_item);

USE FROM topology_schema      —GB/T 16656.42
  (edge_curve,
   edge_loop,
   path,
   vertex_loop,
   vertex_point,
   vertex_shell,
   wire_shell);
(*
```

注 2:以上所引用的模式见 GB/T 16656 的如下部分:

几何模型模式 (geometric_model_schema)	GB/T 16656.42
几何模型 (geometry_schema)	GB/T 16656.42
产品特性表达模式 (product_property_representation_schema)	GB/T 16656.41
表达模式 (representation_schema)	GB/T 16656.43
拓扑模式 (topology_schema)	GB/T 16656.42

4.1 引言

GB/T 16656 的本部分提供了用于表达三维形状的几何和拓扑结构,这些三维形状采用具有 0 维(vertex_shell) 或 1 维(wire_shell)的壳。这种 AIC 通过一种属于 shape_representation 的 shell_based_wireframe_shape_representation 实体来表达。

4.2 基本概念和假设

形状的线框表达是基于 shell_based_wireframe_model 和所需要的几何及拓扑实体的。通过 shell_based_wireframe_shape_representation 所表达的形状是其定义可以由 vertex_shell 或 wire_shell 的集合所满足的形状。线框表达可以是映射到同一坐标空间中其他线框表达的组合。

4.3 基于 aic 壳的线框实体的定义:基于壳的线框形状表达

基于壳的线框形状表达(shell_based_wireframe_shape_representation)是三维 shape_representation,它通过定义隐含体的线框来表达产品的形状或部分形状。它包括全部三维曲线和定义顶点、边及环图形的拓扑实体。

注:采用本 AIC 的应用协议可以保证 shape_representation 实体被实例化为 shell_based_wireframe_shape_representation。

EXPRESSS 描述:

```
ENTITY shell_based_wireframe_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (it <* SELF. items |
        NOT
(SIZEOF(['AIC_SHELL_BASED_WIREFRAME. SHELL_BASED_WIREFRAME_MODEL',
                  'AIC_SHELL_BASED_WIREFRAME. MAPPED_ITEM',
                  'AIC_SHELL_BASED_WIREFRAME. AXIS2_PLACEMENT_3D'] *
            TYPEOF (it)) = 1
       ))) = 0;
  WR2: SIZEOF (QUERY (it <* SELF. items |
            SIZEOF(['AIC_SHELL_BASED_WIREFRAME. SHELL_BASED_WIREFRAME_MODEL',
            'AIC_SHELL_BASED_WIREFRAME. MAPPED_ITEM'] * TYPEOF (it)) = 1
       )) >= 1;
  WR3: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF. items |
         ('AIC_SHELL_BASED_WIREFRAME. SHELL_BASED_WIREFRAME_MODEL'
              IN TYPEOF (it))) |
         NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
                              sbwm\shell_based_wireframe_model. sbwm_boundary |
                    ('AIC_SHELL_BASED_WIREFRAME. WIRE_SHELL' IN TYPEOF (sb))) |
           NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
                               ws\wire_shell. wire_shell_extent |
                 ('AIC_SHELL_BASED_WIREFRAME. EDGE_LOOP' IN TYPEOF (wsb))) |
             NOT (SIZEOF (QUERY (el <* eloop\path. edge_list |
               NOT ('AIC_SHELL_BASED_WIREFRAME. EDGE_CURVE' IN
                     TYPEOF (el. edge_element)))) = 0)
           )) = 0)
         )) = 0)
       )) = 0;
  WR4: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF. items |
         ('AIC_SHELL_BASED_WIREFRAME. SHELL_BASED_WIREFRAME_MODEL'
              IN TYPEOF (it))) |
         NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
                              sbwm\shell_based_wireframe_model. sbwm_boundary |
                    ('AIC_SHELL_BASED_WIREFRAME. WIRE_SHELL' IN TYPEOF (sb))) |
```

```
        NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
                              ws\wire_shell. wire_shell_extent |
          ('AIC_SHELL_BASED_WIREFRAME. EDGE_LOOP' IN TYPEOF (wsb))) |
        NOT (SIZEOF (QUERY (pline_el <*
                          QUERY (el <* eloop\path. edge_list |
            ('AIC_SHELL_BASED_WIREFRAME. POLYLINE' IN
              TYPEOF (el. edge_element\edge_curve. edge_geometry))) |
          NOT (SIZEOF (pline_el. edge_element\edge_curve.
                          edge_geometry\polyline. points) >2)
        )) = 0)
      )) = 0)
    )) = 0)
  )) = 0;
WR5: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF. items |
    ('AIC_SHELL_BASED_WIREFRAME. SHELL_BASED_WIREFRAME_MODEL'
      IN TYPEOF (it))) |
    NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
                          sbwm\shell_based_wireframe_model. sbwm_boundary |
        ('AIC_SHELL_BASED_WIREFRAME. WIRE_SHELL' IN TYPEOF (sb))) |
      NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
                              ws\wire_shell. wire_shell_extent |
        ('AIC_SHELL_BASED_WIREFRAME. EDGE_LOOP' IN TYPEOF (wsb))) |
        NOT (SIZEOF (QUERY (el <* eloop\path. edge_list |
          NOT (valid_wireframe_edge_curve
                (el. edge_element\edge_curve. edge_geometry )))) =0)
    )) = 0)
    )) = 0)
  )) = 0;
WR6: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF. items |
    ('AIC_SHELL_BASED_WIREFRAME. SHELL_BASED_WIREFRAME_MODEL'
      IN TYPEOF(it))) |
    NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
                          sbwm\shell_based_wireframe_model. sbwm_boundary |
        ('AIC_SHELL_BASED_WIREFRAME. WIRE_SHELL' IN TYPEOF (sb))) |
      NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
                              ws\wire_shell. wire_shell_extent |
        ('AIC_SHELL_BASED_WIREFRAME. EDGE_LOOP' IN TYPEOF (wsb))) |
        NOT (SIZEOF (QUERY (el <* eloop\path. edge_list |
          NOT (('AIC_SHELL_BASED_WIREFRAME. VERTEX_POINT' IN
                TYPEOF (el. edge_element. edge_start))
            AND
              ('AIC_SHELL_BASED_WIREFRAME. VERTEX_POINT' IN
                TYPEOF (el. edge_element. edge_end))) )) = 0)
```

```
        )) = 0)
      )) = 0)
    )) = 0;
WR7: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF. items |
      ('AIC_SHELL_BASED_WIREFRAME. SHELL_BASED_WIREFRAME_MODEL'
        IN TYPEOF (it))) |
      NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
                    sbwm\shell_based_wireframe_model. sbwm_boundary |
                ('AIC_SHELL_BASED_WIREFRAME. WIRE_SHELL' IN TYPEOF (sb))) |
        NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
                    ws\wire_shell. wire_shell_extent |
            ('AIC_SHELL_BASED_WIREFRAME. EDGE_LOOP' IN TYPEOF (wsb))) |
          NOT (SIZEOF (QUERY (el <* eloop\path. edge_list |
              NOT ((valid_wireframe_vertex_point
                  (el. edge_element.
                    edge_start\vertex_point. vertex_geometry))
                AND
                (valid_wireframe_vertex_point
                (el. edge_element. edge_end\vertex_point. vertex_geometry)))
          )) = 0)
        )) = 0)
      )) = 0)
    )) = 0;

WR8: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF. items |
      ('AIC_SHELL_BASED_WIREFRAME. SHELL_BASED_WIREFRAME_MODEL'
        IN TYPEOF(it))) |
      NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
                    sbwm\shell_based_wireframe_model. sbwm_boundary |
                ('AIC_SHELL_BASED_WIREFRAME. WIRE_SHELL' IN TYPEOF (sb))) |
        NOT (SIZEOF (QUERY (vloop <* QUERY (wsb <*
                    ws\wire_shell. wire_shell_extent |
            ('AIC_SHELL_BASED_WIREFRAME. VERTEX_LOOP' IN TYPEOF (wsb))) |
          NOT ('AIC_SHELL_BASED_WIREFRAME. VERTEX_POINT' IN
              TYPEOF (vloop\vertex_loop. loop_vertex))
        )) = 0)
      )) = 0)
    )) = 0;
WR9: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF. items |
      ('AIC_SHELL_BASED_WIREFRAME. SHELL_BASED_WIREFRAME_MODEL'
        IN TYPEOF(it))) |
      NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
                    sbwm\shell_based_wireframe_model. sbwm_boundary |
```

```
                ('AIC_SHELL_BASED_WIREFRAME.WIRE_SHELL' IN TYPEOF (sb))) |
            NOT (SIZEOF (QUERY (vloop <* QUERY (wsb <*
                                ws\wire_shell.wire_shell_extent |
                    ('AIC_SHELL_BASED_WIREFRAME.VERTEX_LOOP' IN TYPEOF (wsb))) |
                NOT (valid_wireframe_vertex_point (vloop\vertex_loop.
                    loop_vertex\vertex_point.vertex_geometry))
            )) = 0)
        )) = 0)
    )) = 0;
  WR10: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
        ('AIC_SHELL_BASED_WIREFRAME.SHELL_BASED_WIREFRAME_MODEL'
            IN TYPEOF(it))) |
        NOT (SIZEOF (QUERY (vs <* QUERY (sb <*
                            sbwm\shell_based_wireframe_model.sbwm_boundary |
                    ('AIC_SHELL_BASED_WIREFRAME.VERTEX_SHELL' IN TYPEOF (sb))) |
            NOT ('AIC_SHELL_BASED_WIREFRAME.VERTEX_POINT' IN
                TYPEOF (vs\vertex_shell.vertex_shell_extent.loop_vertex))
        )) = 0)
    )) = 0;
  WR11: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
        ('AIC_SHELL_BASED_WIREFRAME.SHELL_BASED_WIREFRAME_MODEL'
            IN TYPEOF(it))) |
        NOT (SIZEOF (QUERY (vs <* QUERY (sb <*
                            sbwm\shell_based_wireframe_model.sbwm_boundary |
                    ('AIC_SHELL_BASED_WIREFRAME.VERTEX_SHELL' IN TYPEOF (sb))) |
            NOT (valid_wireframe_vertex_point (vs\vertex_shell.
                vertex_shell_extent.loop_vertex\vertex_point.
                vertex_geometry))
        )) = 0)
    )) = 0;
  WR12: SIZEOF (QUERY (mi <* QUERY (it <* SELF.items |
        ('AIC_SHELL_BASED_WIREFRAME.MAPPED_ITEM' IN TYPEOF (it))) |
        NOT ('AIC_SHELL_BASED_WIREFRAME.' +
            'SHELL_BASED_WIREFRAME_SHAPE_REPRESENTATION' IN
        TYPEOF (mi\mapped_item.mapping_source.mapped_representation)
    ))) = 0;
  WR13: SELF.context_of_items\geometric_representation_context.
        coordinate_space_dimension = 3;
END_ENTITY;
(*
```

形式限制：

WR1：在 shell_based_wireframe_shape_representation 中的 item 应该是 shell_based_wireframe_model、mapped_item 或 axis2_placement_3d。

WR2：在 shell_based_wireframe_shape_representation 中，至少有一个 item 应该是 shell_based_wireframe_model，或者是 mapped_item。

WR3：在 shell_based_wireframe_model 中，为 edge_loop 所定义的每条边均应该是 edge_curve。

WR4：形成 shell_based_wireframe_model 的边的每条折线均应该有两个以上不同的点。

WR5：形成 shell_based_wireframe_model 的边的 edge_geometry 应该是 line（线段）、conic（二次曲线）、b_spline_curve（B样条曲线）、offset_curve_3d（三维偏置曲线）或 curve_replica（复制曲线），以其他曲线定义为基础的曲线一向被使用。在 offset_curve_3d 或 curve_replica 的情况下，作为 basic_curve 所引用的曲线应该是上述类型之一。

WR6：被定义为 shell_based_wireframe_model 的边的起点和终点的每个顶点均应该是 vertex_point。

WR7：形成顶点的 vertex_geometry 应该是 cartesian_point 或 point_replica，而 point_replica 应该是另外的 point_replica 或 cartesian_point 的复制，该顶点是用于定义 shell_based_wireframe_model 的 edge_loop 的边的边界。

WR8：用来定义作为 shell_based_wireframe_model 的边界使用的 vertex_loop 的顶点应当是 vertex_point。

WR9：用来定义作为 shell_based_wireframe_model 的边界使用的 vertex_loop 的顶点应该由 cartesian_point 或 point_replica 形成，而 point_replica 应该是其他 point_replica 或 cartesian_point 的复制。

WR10：用来定义作为 shell_based_wireframe_model 中 vertex_shell 的 vertex_shell_extent 使用的 vertex_loop 的顶点应该是 vertex_point。

WR11：用来定义作为 shell_based_wireframe_model 中 vertex_shell 的 vertex_shell_entext 所使用的 vertex_loop 的顶点应该由 cartesian_point 或 point_replica underliad 形成，而且 point_replica 应当是其他的 point_replica 或 cartesian_point 的复制。

WR12：如果在 shell_based_wireframe_shape_representation 中包含 mapped_item，那么 mapped_item 的来源应该是一项 shell_based_wireframe_shape_representation。

WR13：shell_based_wireframe_shape_representation 应该具有等于 3 的 coordinate_space_dimension。

4.4 基于 aic 壳的线框的函数定义

4.4.1 有效线框的边曲线

valid_wireframe_edge_curve（有效线框的边曲线）函数用于判断一条输入曲线在表达由拓扑有界线框所定义的形状时是否有效。

EXPRESSS 描述：

```
FUNCTION valid_wireframe_edge_curve (crv     : curve) : BOOLEAN;
  —check for valid basic curve types

  IF SIZEOF (['AIC_SHELL_BASED_WIREFRAME. LINE',
                          'AIC_SHELL_BASED_WIREFRAME. CONIC',
              'AIC_SHELL_BASED_WIREFRAME. B_SPLINE_CURVE',
              'AIC_SHELL_BASED_WIREFRAME. POLYLINE'] * TYPEOF (crv)) = 1
    THEN RETURN (TRUE);
```

```
  ELSE
  —recursively check for valid basic curves for curve_replica
    IF ('AIC_SHELL_BASED_WIREFRAME. CURVE_REPLICA') IN TYPEOF (crv)
      THEN RETURN (valid_wireframe_edge_curve
                          (crv\curve_replica. parent_curve));
    ELSE
      —recursively check for valid basis curves for offset_curve
      IF ('AIC_SHELL_BASED_WIREFRAME. OFFSET_CURVE_3D') IN TYPEOF (crv)
        THEN RETURN (valid_wireframe_edge_curve
                          (crv\offset_curve_3d. basis_curve));
      END_IF;
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION;
(*
```

变量定义：

crv：被检查的输入曲线。有效曲线是 line、conic、b_spline_curve、polyline 和 offset_curve_3d，或者是 curve_replica。当有效曲线是 offset_curve_3d 或 curve_replica 时，被引用为 basis_curve 或 parent_curve 的曲线应该是有效曲线。

4.4.2 有效线框的顶点点

valid_wireframe_vertex_point(有效线框的顶点点)函数用于判断一个输入点在表达由拓扑有界线框所定义的形状时是否有效。

EXPRESSS 描述：

```
*)
FUNCTION valid_wireframe_vertex_point (pnt     : point): BOOLEAN;
  —check for valid basic point types
  IF ( 'AIC_SHELL_BASED_WIREFRAME. CARTESIAN_POINT' IN TYPEOF (pnt))
    THEN RETURN (TRUE);
  ELSE
    —recursively check for valid basic point types as parents for a
    —point_replica

    IF ('AIC_SHELL_BASED_WIREFRAME. POINT_REPLICA') IN TYPEOF (pnt)
      THEN RETURN (valid_wireframe_vertex_point
                          (pnt\point_replica. parent_pt));
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION;
(*
```

变量定义：

pnt：被检查的输入点。有效点是 cartesian_point 或 point_replica。当有效点是 point_replica 时，parent_point 也应该是有效点。

*)

END_SCHEMA;

(*

附 录 A
（规范性附录）
实体短名

表 A.1 提供了在 GB/T 16656 本部分的 EXPRESS 短表中所规定的实体短名。有关短名的使用要求见 GB/T 16656 的实现方法。

表 A.1 实 体 短 名

实 体 名	短 名
SHELL_BASED_WIREFRAME _SHAPE_REPRESENTATION	SBWSR

附 录 B
（规范性附录）
信息对象注册

B.1 文档标识

在开放系统中，为提供信息对象的无二义标识，规定本部分的对象标识符为：

{iso standard 10303 part(502) version(1)}

本标识符的含义在 GB/T 16262 中进行了定义，并且在 GB/T 16656.1 中进行了描述。

B.2 模式标识

在开放系统中，为提供 aic_shell_based_wireframe_shema 的无二义标识，规定 aic_shell_based_wireframe_shema 模式的对象标识符为（见第 4 章）：

{iso standard 10303 part(502) version(1)object(1) aic_shell_based_wireframe_shema (1)}

本标识符的含义在 GB/T 16262 中进行了定义，并且在 GB/T 16656.1 中进行了描述。

附 录 C
（资料性附录）
EXPRESS-G 图

图 C.1 至图 C.6 对应于采用 GB/T 16656.11 接口规范的第 4 章中简表所产生的 EXPRESS 描述。这些图使用 EXPRESS 语言的 EXPRESS-G 图的符号表示法。在 GB/T 16656.11 的附录 D 中定义了 EXPRESS-G。

注 1：选择类型 geometric_set_select、transformation、trimming_select、reversible_topology、vector_or_direction 和 wireframe_select 与符合 GB/T 16656.11 隐式接口规则的 AIC 扩展列表相连接。GB/T 16656 本部分的其他实体不使用这些选择类型。