prostep IVIP

Recommendation

prostep ivip Recommendation

ECAD/MCAD Collaboration
Implementation Guidelines

prostep ivip Documentation

Referring to PSI 5

ECAD/MCAD Collaboration

Implementation Guidelines

Version 4.5, 01/09/2020

Status: released

Abstract

This Implementation Guideline accompanies the prostep ivip PSI5 Recommendation for ECAD/MCAD-Collaboration.

Disclaimer

Prostep ivip Certification Guidelines (PSI Certification Guidelines) are available for general use. Anyone using these recommendations is responsible for ensuring that they are used correctly.

This PSI Certification Guidelines gives due consideration to the prevailing state-of-the-art at the time of publication. Anyone using PSI Certification Guidelines must assume responsibility for his or her actions and acts at their own risk. The prostep ivip Association and the parties involved in drawing up the PSI Certification Guidelines assume no liability whatsoever.

We request that anyone encountering an error or the possibility of an incorrect interpretation when using the PSI Recommendation should contact the prostep ivip Association (psi-issues@prostep.com) immediately so that any errors can be rectified.

Copyright

I. All rights on this PSI Documentation, in particular the copyright rights of use and sale such as the right to duplicate, distribute or publish the Documentation remain exclusively with the prostep ivip Association and its members.

II. The PSI Documentation may be duplicated and distributed unchanged, for instance for use in the context of creating software or services.

III. It is not permitted to change or edit this PSI Documentation.

IV. A suitable notice indicating the copyright owner and the restrictions on use must always appear.

Acknowledgment

Our thanks go to all the companies and their staff who were actively involved in drafting this recommendation and for the many constructive suggestions received. The following companies and research institutes were involved:

Cenit, Continental, Delphi, :em engineering methods, Fern Universität Hagen, Fraunhofer IPK, Mentor Graphics, Parametric Technology, PDTec, PROSTEP AG, Siemens PLM Software, Universität Karlsruhe, xPLM Solution, Cadence, CADCAM Group, Dassault Systèmes.

Contents

Table of Contents

Figures

Tables

1 General

1.1 Objectives of the Implementation Guidelines

The main objective of this Implementation Guideline is to help those concerned with the implementation of ECAD/MCAD collaboration using the IDX data exchange format. This includes customization of software, implementation of services as well as definition of ECAD/MCAD processes, including release processes.

These Implementation Guidelines supplement the ECAD/MCAD Recommendation which is available as both a User Version (for users of the format) and Developer Version (aimed at developers writing code to read and write the format).

These Guidelines are designed to give a clear overview of the aims of the format, how it works and how to use it to describe key PCB features. More detailed information about the technical specification of the format can be found in the Recommendation documents.

This Guideline also explains how to map typical test cases into the IDX schema in order to illustrate the ECAD/MCAD approach for most relevant use cases. The test cases described in the document will be checked in the certification process.

1.2 Document structure

This document is structured to introduce the IDX schema in a logical sequence, starting with an introduction to the basic workflows used (section 3), followed by an introduction to the basic 'Item' used for collaboration in section 4.

Section 5 describes the various message/file types used by IDX, whilst sections 6 and 7 show how to model 2D/3D geometry and give examples of how to describe common PCB objects, starting from simple objects before moving to more advanced objects such as flexible boards.

The related XML files can be found in appendix B of the Implementation Guide.

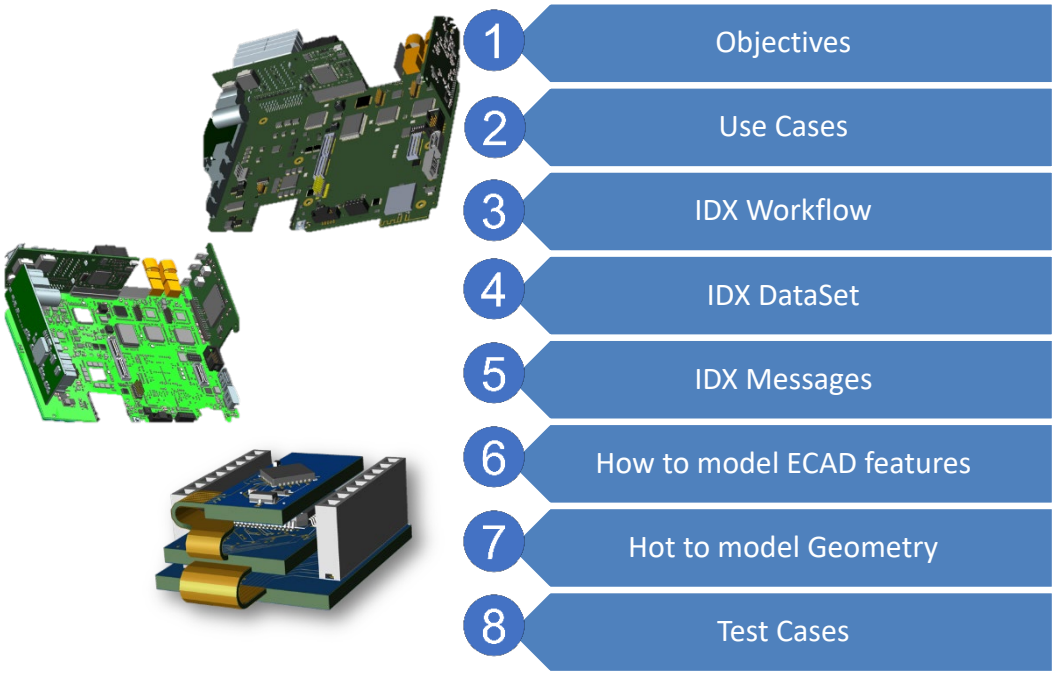


Figure 1 Structure of implementation guidelines

2 Overview of use cases

The recommendation describes the data model and resulting XSD structure to be used to satisfy use cases relevant within a collaborative development of a board design. However, in some cases the recommendation allows for different ways of describing an aspect of the design. This means that although the structure of the XML file is specified, the content may vary since terms for describing the content are not fixed. The objective of these implementation guidelines is specifying the terms in a more detailed way to provide more detailed information to the implementer. Section 2.1 specifies the use cases relevant in the context of implementation. Section 2.2 specifies a list of test cases in the context of the use cases. These test cases cover the most important part of an ECAD/MCAD implementation.

2.1 Use cases covered within implementation guidelines

The implementer guidelines describe more in detail how to support the relevant use cases. Figure 2 below shows the use cases to be detailed within these implementation guidelines:

- Use Case 1 (UC-1): Definition of board baseline
- Use Case 2 (UC-2): Change of placement locations
- Use Case 3 (UC-3): Replacement of components
- Use Case 4 (UC-4): Panelization design review

From a data model (and interface) point of view there is no need to distinguish between “Placement under electrical constraints” and “Placement under mechanical constraints”. Furthermore the “Change of board elements” is either “Change of placement locations” or “Replacement of components”. For detailed description of the use cases see PSI5-ECAD/MCAD Collaboration. Use Case 1 (Definition of board base line) is precondition for Use Cases 2-4. This Guideline gives explains based on representative test data how to how to use the IDX schema / data model. This is an extract of the test cases specified in the certification guideline. This will be described in the following chapters.

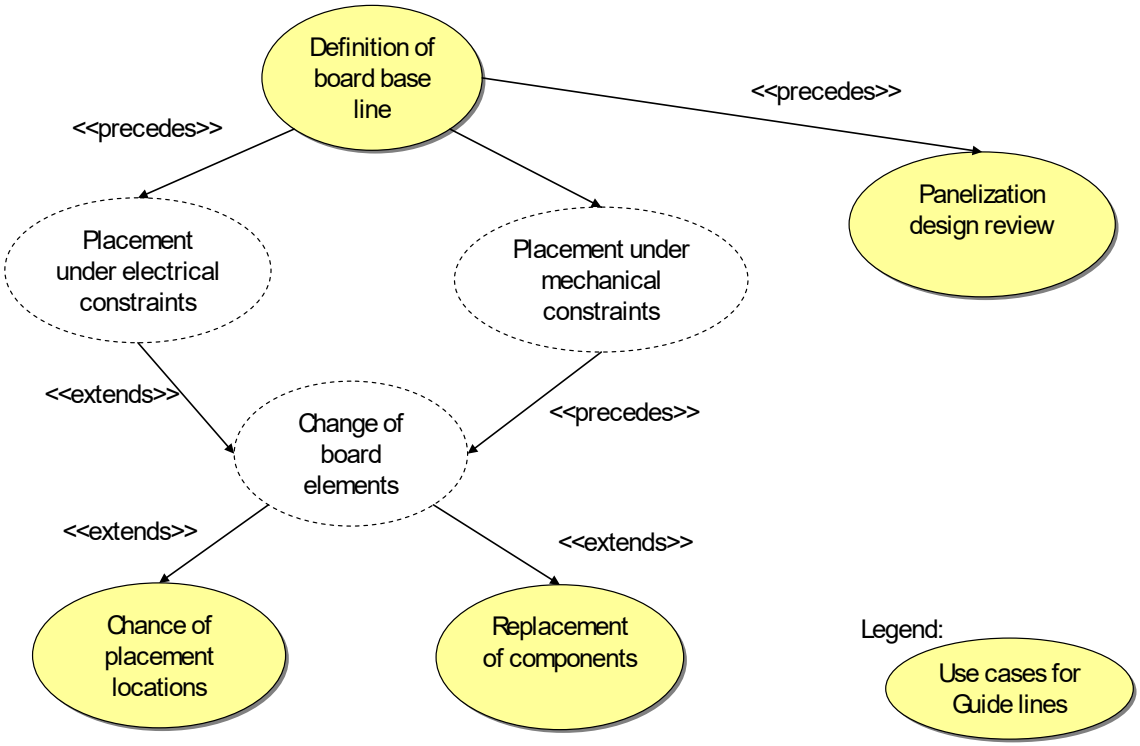


Figure 2: Use Cases covered within implementer guidelines

2.2 Test cases

Based on the use cases to be supported a set of test cases is specified which covers the IDX conformance class 1 specified in conformance guidelines. These test cases are listed in the table below. A more detailed description can be found in chapter 6.

Test case ID	Description	Section in implementer guide	Covered Use Cases (ID)
TC 1	**Definition of board baselines**		
TC-1-1	Definition of initial baseline by supporting different kinds of “explicit” shape representation.	8.1	**UC-1**
TC-1-2	Definition of initial baseline which contains the main item types for collaboration.	8.2	**UC-1**
TC 2	**Execute changes**		
TC-2-1	Addition of new items (Change 1)	8.3	**UC-2, UC-4**
TC-2-2	Removal of items (Change 2)	8.4	**UC-2, UC-4**
TC-2-3	Movement of items (Change 3)	8.5	**UC-2, UC-4**
TC-2-4	Verify modification of items (Change 4)	8.6	**UC-2, UC-3, UC-4**
TC-2-5	Verify replacement of items (Change 5)	8.7	**UC-3, UC-4**

Table 1: Test cases for certification process

3 Introduction to the IDX protocol

The prostep ivip recommendation specifies an XSD schema for describing data to be exchanged between ECAD/MCAD systems as well as a WSDL protocol which describes the method of communication between them. Both the XSD schema and WSDL file are available as a zip file.

Whilst any programs supporting IDX should rigidly adhere to the current XSD schema, the workflow currently described in the WSDL isn't in widespread use, so it's recommended that the workflow information in this guideline is used as a basis for development instead.

The IDX protocol can be used for ECAD/MCAD collaboration use cases involving either synchronous or asynchronous collaboration in either a 1:1 configuration or a 1:n configuration (see Figure 3 below). Most implementations of the IDX protocol to date have implemented a synchronous file-based 1:1 bi-directional ECAD to MCAD workflow.

IDX messages are defined in such a way that they are easy to transfer to other technologies. In WSDL, they are defined as operations. They can however also be understood as function calls involving a shared library, as a call involving remote procedures, or as messages in a message queue. Therefore, no mechanisms for receiver addressing have been defined in the schema since these would have to be made available by the underlying communication layer, e.g. by calling a remote procedure on precisely this remote system or by sending the message to this receiver in the message queue. It is also possible to embed the messages in XML files with an *EDMDDataSet* as the root element.

The most common method of communication used currently for sending and receiving IDX messages is via XML files. These guidelines contain information about best practices when using files to communicate IDX messages.

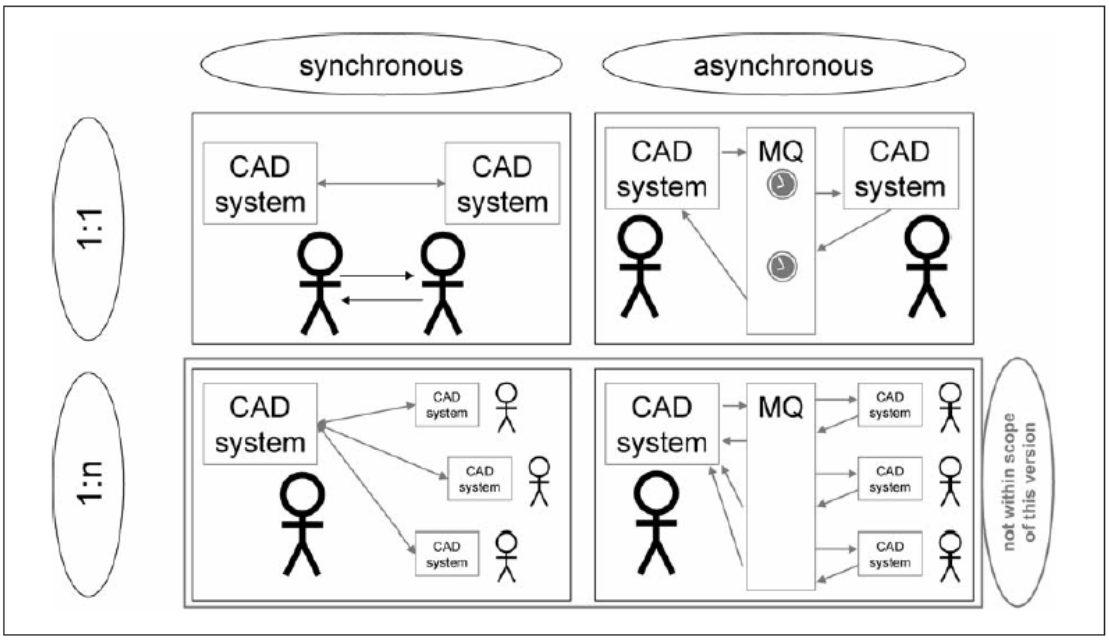


Figure 3: Types of communication

The IDX protocol supports the following three message types:

- **SendInformation:** Used to send a 'Baseline' representing the initial state of the PCB from ECAD to MCAD or vice-versa. This ensures both ECAD and MCAD are using the same dataset before any subsequent changes are sent by either party.
- **SendChanges:** Used to send one or more proposed design changes from ECAD to MCAD or vice-versa (after the initial Baseline has been sent). The SendChanges message is also used to send a Response to say if those proposed changes have been accepted or rejected.

- **RequestForInformation:** Used for one party to request information about the current state of an item or items from others. The state can include information such as the item's ID, shape, or owner. This message type is not currently used by most implementations of IDX.

For more details about each message type see chapter 6 (IDX protocol) of the Recommendation.

3.1 Typical Workflows

3.1.1 Basic IDX Workflow

Most file-based implementations of IDX utilize the following basic workflow:

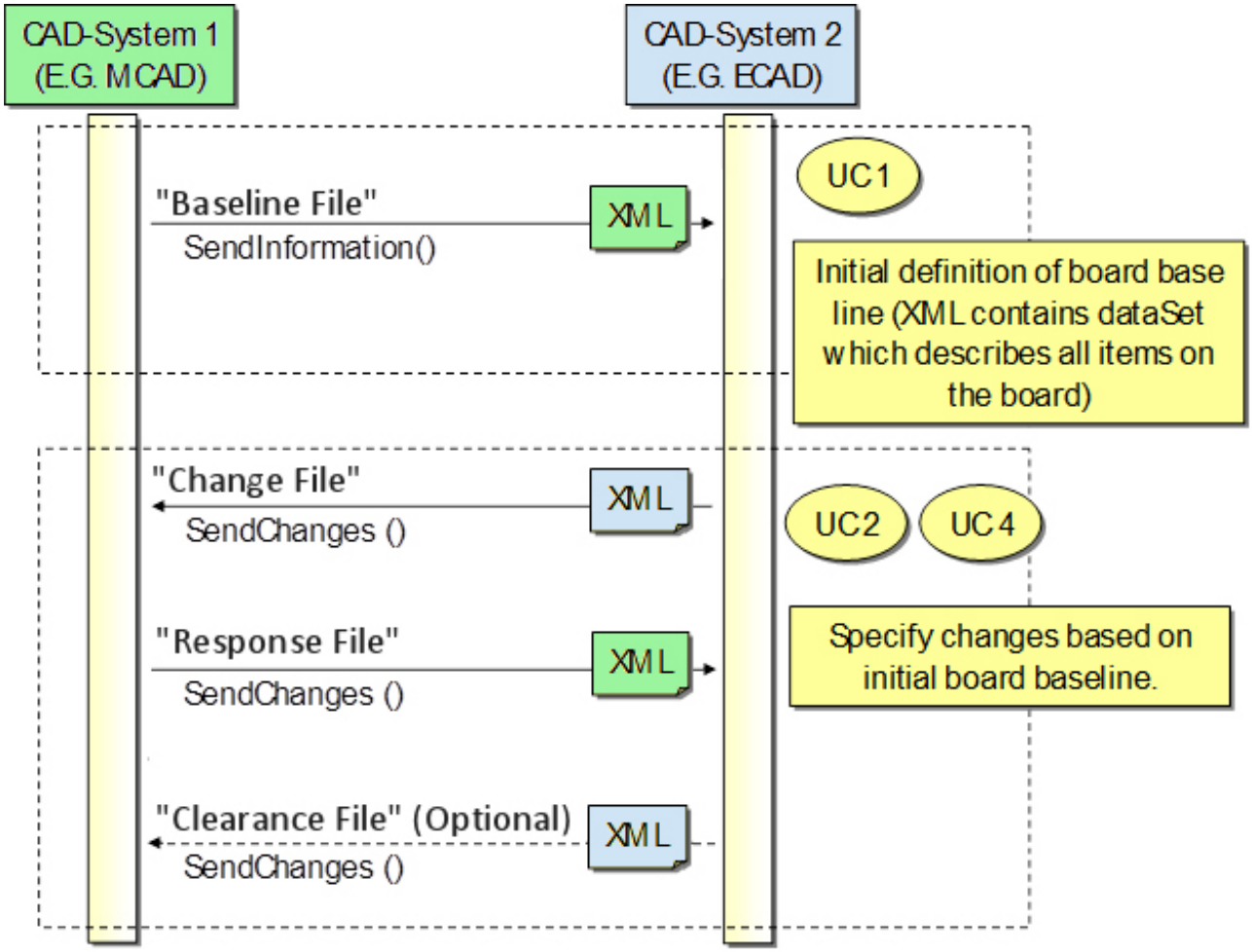


Figure 4: Typical Workflow

1) **Send Baseline**: *User A* sends *User B* a *Baseline file* (of type *EDMDProcessInstructionSendInformation*) containing the complete design at that stage. User B's system must automatically accept the entire contents of the file. This ensures both systems are synchronized with the same initial dataset.

2) **Send Proposed Changes:** Either user may wish to change one or more aspects of the design, such as adding, removing, or moving components or changing the shape of the board. Assuming the changes are made by User B in this case, User B sends their proposed design changes to User A in a *Change file* message of type *EDMDProcessInstructionSendChanges.*

3) **Accept or Reject Proposed Changes:** User A will receive and review the changes from User B and can choose to accept or reject some or all of them. They will then send back a *Response File* message of type *EDMDProcessInstructionSendChanges* with the items proposed for change annotated with a *<pdm:accept>* attribute where a value of *true* indicates the proposed change was accepted, and *false* indicates it was rejected. If User B rejects a change proposed by User A, then User A's system must undo the design change so both systems are again synchronized. Each user can continue steps 2 and 3 until the design is complete.

3.1.2 Enhanced IDX Workflows

Some EDMD implementations add additional steps to this workflow in order to address specific challenges including additional Baseline and 'Clearance' file as described below:

3.1.2.1 Additional Baseline Files

Normally a Baseline file is only sent once per design in order to ensure both parties start collaboration with the same dataset. However additional Baseline files can be sent for two reasons, either to 're-Baseline' the design, or to send an additional 'Copper Baseline' file.

1) '**Re-baselining':** Depending on the implementation method, it could be possible for the parties involved in ECAD/MCAD collaboration out of sync with each other. A proposed change could be rejected but not undone correctly for example or an unexpected crash in either system could potentially cause the ECAD and MCAD versions of the design to diverge. In this situation a new Baseline file can be sent by either party to the other to reset the design back to a known point that will be the same in both ECAD and MCAD. Proposed changes can then be made against the new baseline.

2) **'Copper' or supplementary Baselines:** Sometimes it is desirable to send additional design information from one party to the other which is intended to supplement the detail of the design, but not to form the basis of future collaboration. One example of this is a 'Copper Baseline'. This is an additional baseline file sent from ECAD to MCAD containing the geometry of the copper routing/traces on the board. As this file is likely to be very large and the entities within it will not form the basis for collaboration it can be sent as an additional Baseline file.

It is important that the receiving system recognizes the distinction between these two Baseline types. The first is intended to completely replace the current design; the second is intended to supplement it.

Currently there is no way to automatically differentiate between the baseline types. Some implementations assume that any baseline file without a defined board outline definition is a "Copper" or supplementary baseline file, but that cannot be guaranteed. It is suggested that implementations of IDX prompt the user to choose whether to either supplement or replace their data when receiving a supplementary baseline.

3.1.2.2 Clearance Files

Some implementations of IDX also send "Clearance Files". The purpose of a Clearance File is to act as a confirmation that an Accept or Reject instruction has been received and acted on – a bit like a 'Read Receipt' for an email. This workflow is not currently widely supported.

Without a Clearance File a typical workflow could be:

1. A sends B a **Baseline File**. This is automatically accepted and both A and B are in sync.
2. B sends A a **Change file** proposing a design change.
3. A sends back a **Response File** rejecting B's proposal.

When B receives the Response File it should undo its proposed change, reverting its design back to the Baseline state in this case. However, what if B never received the Response file, or received it and did not apply it to their design? In this case the design change would still exist for user B but not user A, hence the two copies of the design would have diverted.

To prevent this situation occurring, a 4th "Clearance" stage can be added to the above workflow as follows:

1. A sends B a **Baseline File**. This is automatically accepted and both A and B are in sync.
2. B sends A a **Change file** proposing a design change.
3. A sends a **Response File** rejecting B's proposal.
4. B receives and applies the Response File, resetting their version of the design back to the same state as A and then sends A a **Clearance File to** confirm the Response File was applied.

The structure of a Clearance File is the same as a Change or Response file, but each Item is annotated with a ‘pdm:ClearanceState‘ attribute rather than a ‘pdm:Accept’ attribute.

For example (note most of the file removed for brevity):

```
<foundation:EDMDDataSet>
    <foundation:Header>
    </foundation:Header>

    <foundation:Body>
        <foundation:Item id="ITEM_1">

        <pdm:ItemType>assembly</pdm:ItemType>

            <pdm:ClearanceState>

                <pdm:StateName/>

                <pdm:Accept>true</pdm:Accept>

            </pdm:ClearanceState>

         </foundation:Item>
    </foundation:Body>

    <foundation:ProcessInstruction xsi:type="computational:EDMDProcessInstructionSendChanges">

    </foundation:ProcessInstruction>

</foundation:EDMDDataSet>
```

4 Representation of an *EDMDDataSet* in XML

Each type of IDX message (*SendChanges, SendInformation, or RequestForInformation*) follows the same basic structure and is defined as an *EDMDDataSet*.

The body of the *EDMDDataSet* structure describes the items' (and item instances) shapes and user properties, as well as optionally proposed changes to those items. The underlying IDX schema elements are located in *EDMDSchema.pdm*, *EDMDSchema.d2* and *EDMDSchema.foundation*.

The *EDMDDataSet* section consists of a "Header" section and "Body" section. The "Header" section provides general information about the *EDMDDataSet* such as the user and system that created it, the version of the software that created it and the global units used (see Figure 5). The "Body" section contains the description of the items and their shapes. In the following the instantiation of the body section will be described.

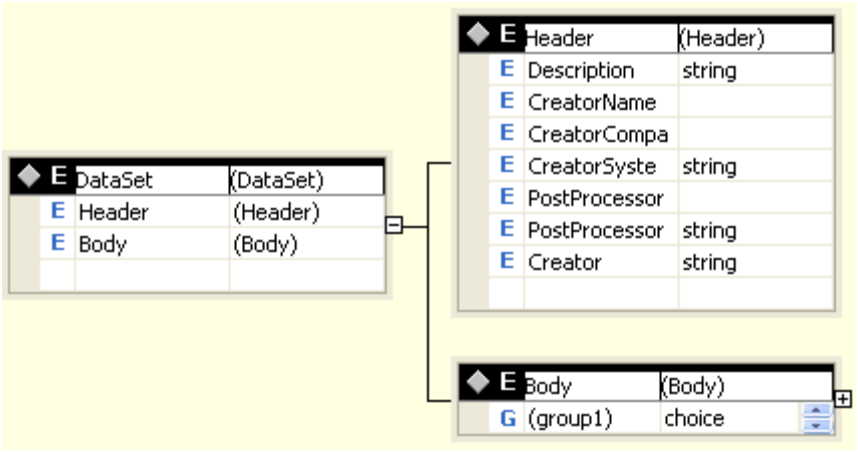


Figure 5: Structure of DataSet section

The "Body" section contains the description of the items, item instances and their shapes that describe either the entire PCB (for a Baseline file), or a proposed change to an item or items (for a SendChanges or 'Change' file).

The type of each item (e.g. Board, Component, Hole, Keepout etc) can be defined either by specifying a geometryType attribute on the item (the preferred method for IDXv4.0), or by instantiation of a subclass and type attributes depending on the type of element (the method used prior to IDXv4.0)

The shape of an item (or item instance) can be described as (see Figure 6)

- "Implicit" by an external file or
- "Explicit" by approximation of the shape by "translation bodies". A "translation body" could be described by a curve and a height (represented by upper and lower bounds). This concept is called "2.5 D geometry"

Most existing implementations of IDX explicitly describe the shapes of geometry (see Section 7).

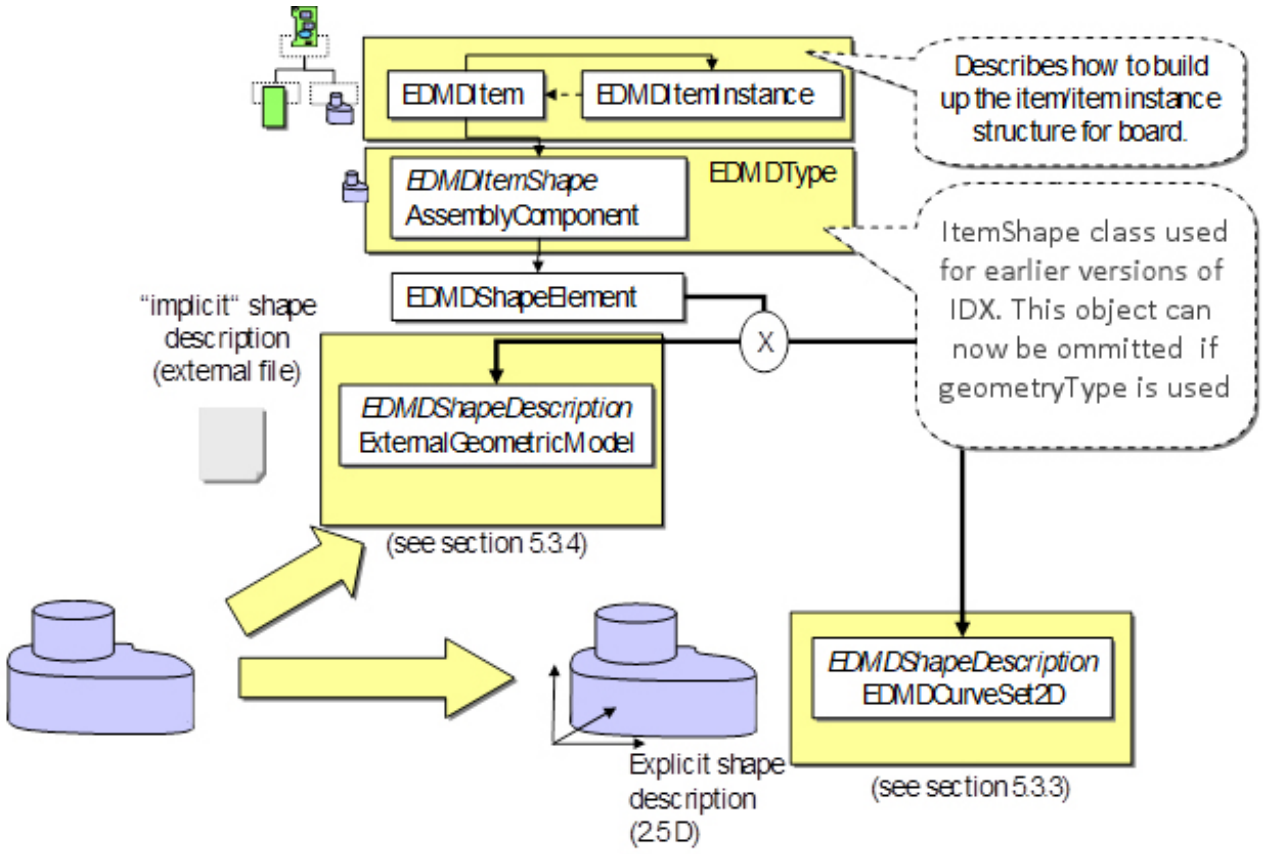


Figure 6: Overview on section giving advice on how items should be instantiated in IDX schema

4.1 Representation of an Item

The *EDMDItem* (and Item Instance) is the main concept of ECAD/MCAD collaboration. It describes any object which could be subject of an ECAD/MCAD collaboration. Figure 9 shows the XSD schema for an item. The UML oriented visualisation of this situation is shown in the Recommendation documents.

An item describes the shape, properties etc. of a collaboration object such as PCB component or keep-out area. An item may be identified differently in each context (here: CAD system).

There are two ways to identify an item:

- Identifier: An identifier must be unique within the context (E.g. ECAD system)
- PackageName: A package name must be unique in within a list of elements.

An item could be an assembly or single component. The item instance represents an occurrence of the item. If an item consists of instances of other items, it contains an “ItemType” attribute of type “assembly” which will be otherwise set to “single”. For example, a PCB board could be seen as an assembly which consists of cut-outs, mounting holes etc. The position of the instance (e.g. mounting hole) within the assembly (E.g. PCB) is described by Transformation (*EDMDTransformation*) of the item instance. The attribute “BaseLine” indicates if the Item belongs to the baseline.

All occurrences of an item are called *EDMDItemInstance* in the IDX schema. Each item occurrence (or instance) refers to its type (of *EDMDItem*).

IDX deals with changes on a board. Each change may apply to an item or item instance (e.g. movement of an instance of a component). In the IDX schema changes are described based on *EDMDItem*.

One change message may consist of several suggested changes. The collaboration partner may accept some and reject the other. In order to support the functionality of ECAD or MCAD systems, each item instance is related with an *EDMDItem* (of type assembly). This item is used to express changes on the related instance (item instance).

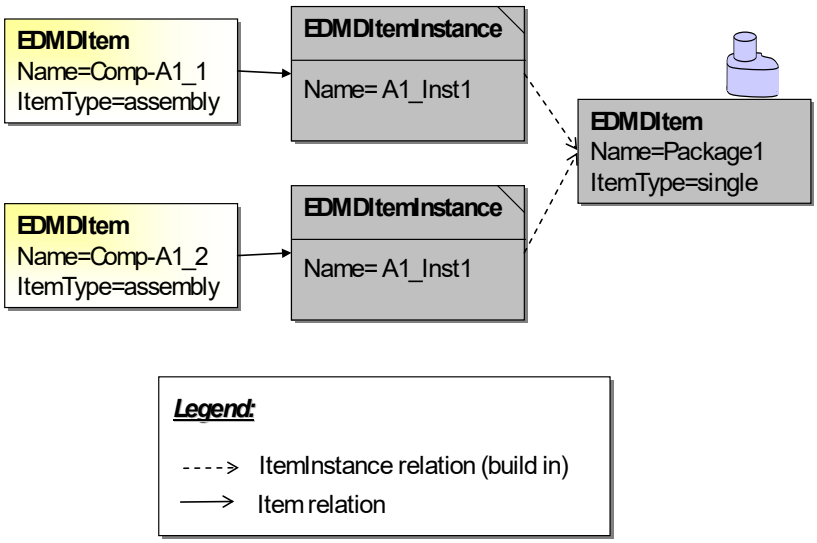


Figure 7: Representation of two instances of a PCB package in IDX

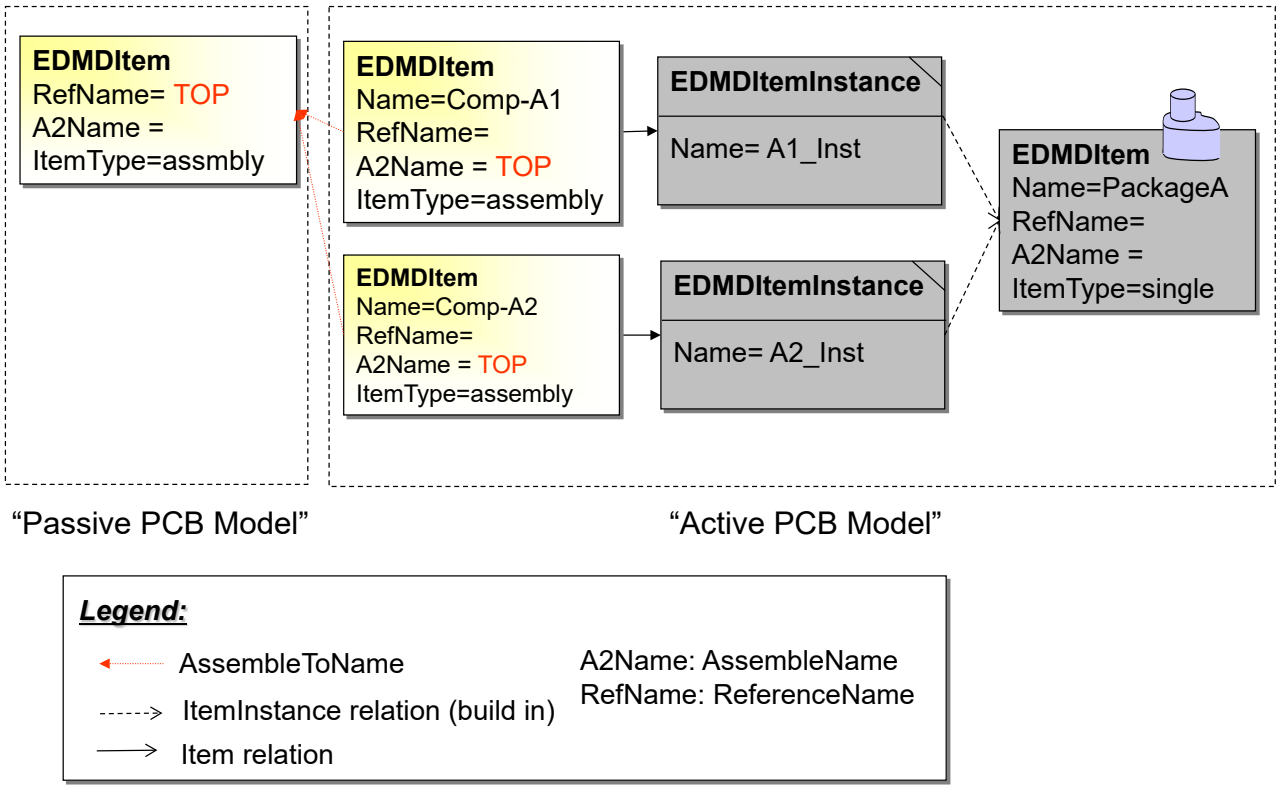


Figure 8: Representation of instances of board components

The reference to the shape (*EDMDItemShape*) of the item is realized by an IDREF reference.

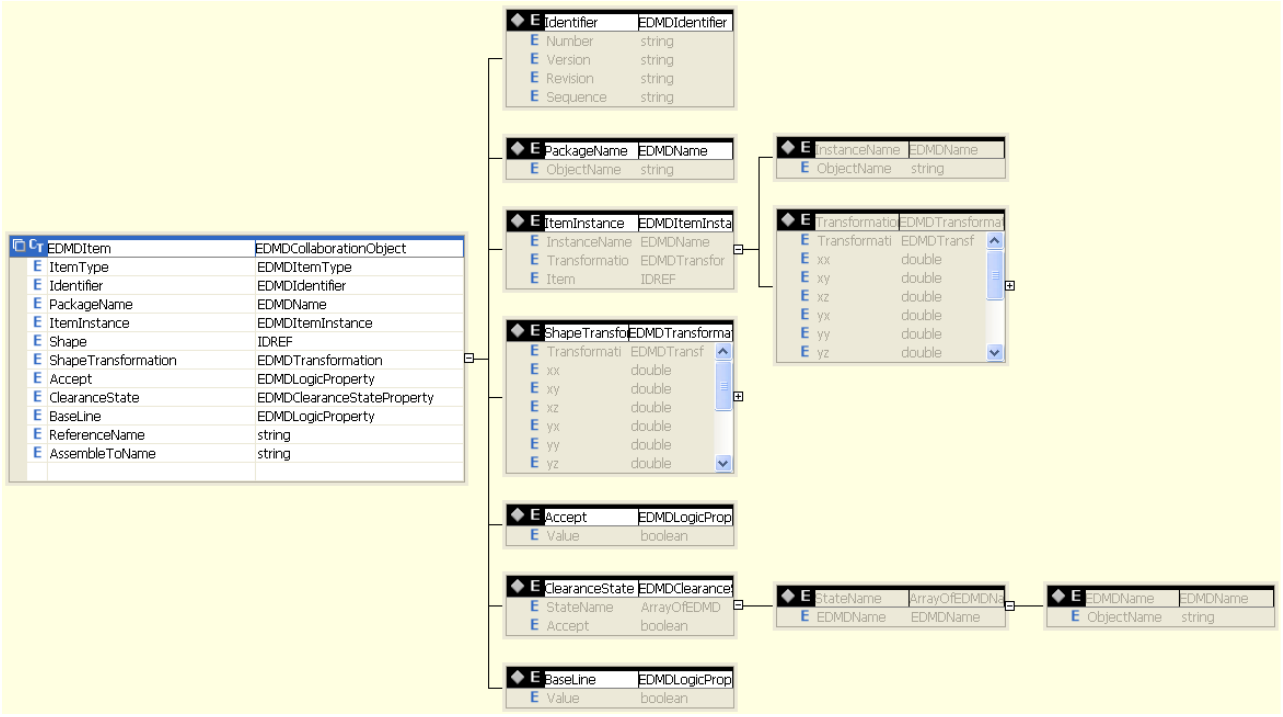


Figure 9: XSD schema for an item

4.1.1 Defining an Item's position in the z-Axis

Objects on the PCB can be represented using either the "Active" or "Passive" PCB model as shown above.

The "Active" model is used to describe the position of an item in terms of its absolute location on the z-axis as described above. While this approach is very straightforward, the disadvantage of this approach is that items' z-axis positions will not automatically change as a result of other design changes. An increase in the board thickness would not automatically change the z-axis positions of the components on the top of the board for example, leaving them partially recessed into the new thicker board.

For this reason, it is recommended that the positions of items are defined relative to a layer surface rather than using absolute z locations wherever possible using the "Passive" model:

The "Passive" model describes the position of components or other items relative to a board or layer surface.

A component may be located on the top or bottom of a board surface or layer or offset from that surface to represent placement within a board cavity for example. The advantage of defining objects' positions relative to a layer surface is that if layer thicknesses are changed the z-axis positions of any items defined relative to those layers will also change.

Items within this model use the attribute "ReferenceName" to define a reference that other items can use to define their positions relative to. For example, in Figure 9 an *EDMDItem* representing the surface of the board has the *ReferenceName* of "TOP". Two top-level items representing instances A1 and A2 of "Package A" reference that *EDMDItem* using the attribute "AssembleToName".

In both cases, "z=0" is assumed to be the surface on the underside of the board to which components would be mounted e.g. the BOTTOM of the board. So, for example, a simple board of 1.6mm thickness would span a range from z=0 to z=1.6. Components on the underside or BOTTOM of the board would be inserted "upside-down" on the z=0 plane, components on the TOP of the board would be inserted "right way up" on the z=1.6 plane.

If the board is defined in terms of a layer stack-up, then it is possible some layers could have a negative z range in this scenario. Soldermask or silkscreen on the underside of the board could span a z range from z= -0.01mm to z=0 mm for example as these layers are further away from the underside mounting surface of the board.

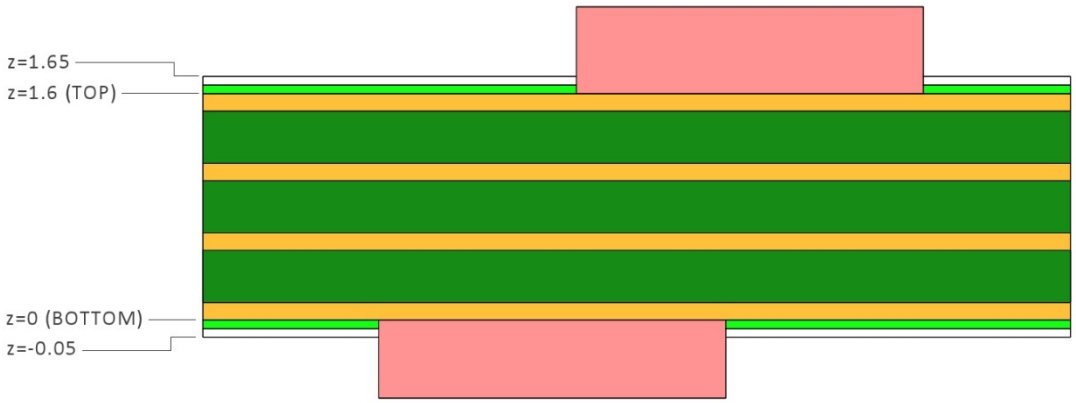


Figure 10: Cross section of board showing Z range values

The diagram above shows a simplified view of a board cross section as defined in above. The 'BOTTOM' of the board is defined as the mounting surface of components on the underside of the board and is at z=0, but some layers such as silkscreen and soldermask on the underside of the board could have a negative z-range as they're below the component mounting surface.

Similarly, in this example components on the TOP mounting surface are at z=1.6, but the top Silkscreen and Soldermask is above the mounting surface with a higher z-range.

4.1.1.1 Locating Item Instances using Transformations

There are two transformation types available in the IDX schema: 2D and 3D. Both are described by standard identity matrices commonly used in 3D CAD and computer graphics.

When a 2D transformation is used the parameters describing z offset and rotation (xz, yz, zz, tz) are null.

Instances that are located with a 2D transformation can also have an optional "zOffset" property. This combination can be useful when positioning a component instance relative to a surface of the board for example: The 'AssembleToName' property specifies a layer or board surface and hence a z value, the 2D transform describes the x,y position and any rotation around the z-axis and then the optional *zOffset* can be used to position the component in z above or below the surface specified by 'AssembleToName' if required.

This approach is more flexible than using a 3D transform to describe the position of an Instance that is likely to move. Whilst a 3D transform describes a fixed point in space, the "2D+zOffset" approach describes a position relative to a surface, so if that surface is moved (perhaps as a result of a change in board thickness), the instance will also move.

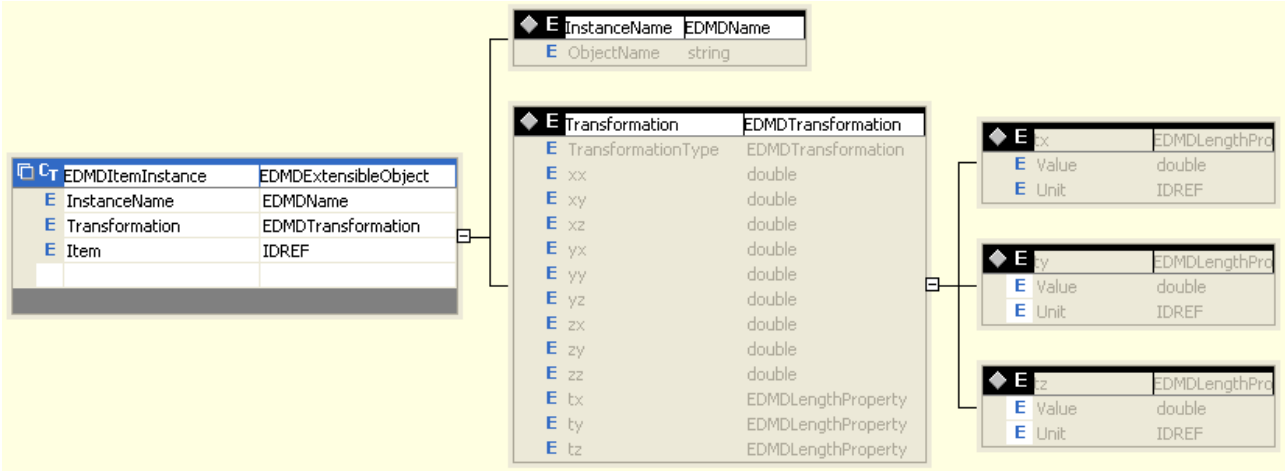


Figure 11: Description of an item instance

The instantiation of items (and item instances) can be shown based on PCB board with PCB components and design features like mounting holes (see Figure 12). The PCB board contains a mounting hole (MH1), a cut-out (CO1) and a keep-out area (KO1). The following PCB components are placed on the board: two instances (A1, A2) of a microcontroller (item A) and one instance (B1) of a capacitor (B).

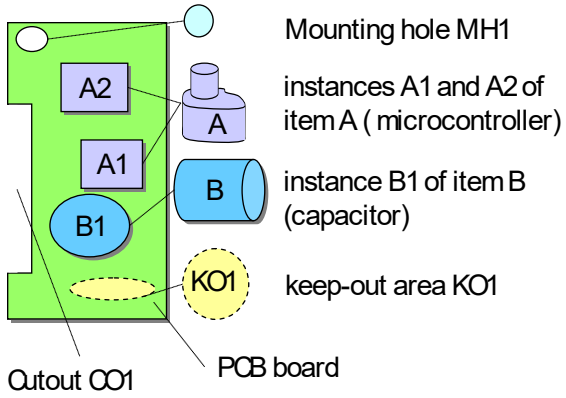


Figure 12: Example PCB board with components and design features

While there are several ways to instantiate the example board according to IDX, the implementation guide suggests specifying an assembly (PCB), which contains all elements. There are two options for structuring the assembly which are considered as suitable and should be supported:

- Define all elements as subparts of the assembly PCB. This option is illustrated in Figure 13.
- Group all non-PCB components in an additional (abstract) assembly “Board”, which consists of the plain board and design features. This option is recommended by this guideline.

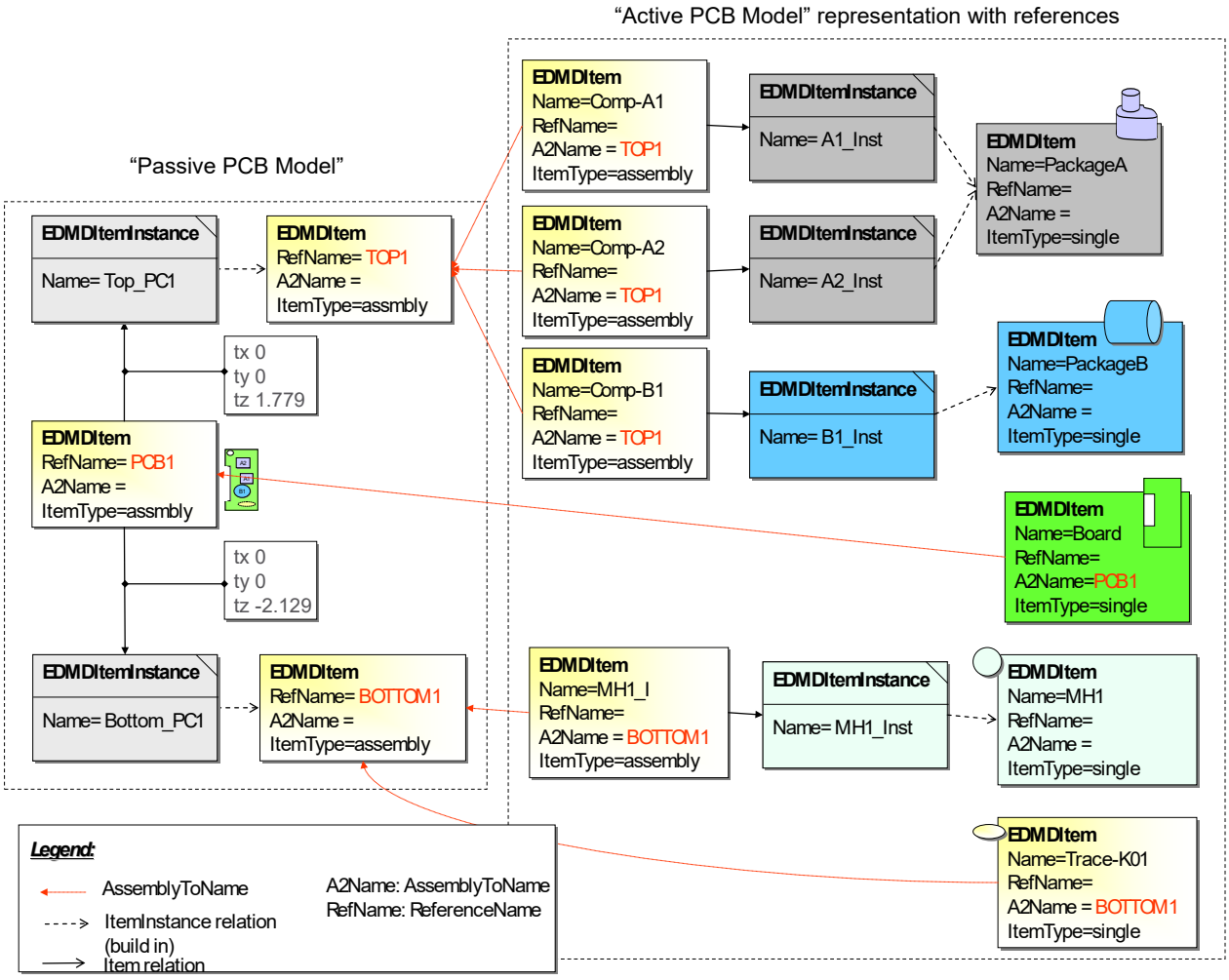


Figure 13: Possible instantiation of a PCB board

Each item instance needs to describe its position regarding its assembly. This is done by the class *EDMDTransformation*. The description of the shape of an item is realized by an instance of the abstract class *EDMDShape*. Which class should be use for certain kinds of items is described in section 0. In order to keep the diagram "readable" this is only shown for A1 and MH1.

4.1.2 Using the correct classification for an item

Each PCB feature type supported by the IDX schema (board, component, hole, keep-out etc.) is described as an 'Item' of type *EDMDItem*. In order to determine which feature each item represents the IDX recommendation requires each item to be further classified by two or three "parameters" (see Figure 14). These additional parameters and child objects allow the system reading the IDX data to determine what PCB feature an Item describes. Finally, the shape of each feature is described by an *EDMDItemShape*.

For IDXv4.0 it was agreed there were some disadvantages with this method of describing PCB features: Every feature requires a lot of XML to describe it, and in some cases the properties of one or more of each item's children need to be analysed to determine the PCB feature type it describes.

To reduce the complexity of how each PCB feature is described, IDXv4.0 introduces a simplified way of describing features by adding an optional 'geometryType' property to each Item. Specifying an Item's *geometryType* allows the user to simplify the subsequent description of each feature.

This guide shows how to describe each PCB feature using both methods; the 'original' method supported by all versions of IDX up to and including version 4.0 and the 'simplified' method only supported from version 4.0 onwards.

Systems reading IDXv4.0 files should be capable of reading both the 'original' and 'simplified' methods of describing PCB features but can use either method to describe features when writing files. It is recommended the 'simplified' method is used to reduce the size and complexity of the IDX files. The 'original' method of describing PCB features will also be depreciated in a future version of the IDX schema.

4.1.2.1 How PCB features are described using the 'original' method

Each item is described by two or three child objects and parameters. For example, a keep-in area would be described by an *EDMDItem* of type assembly (representing the keep-in instance), this in turn references an *EDMDItem* of type single (representing the keep-in itself). Rather than directly referencing a shape, this in turn references an object of type *EDMDKeepIn* (telling us the item is a keep-in). The *EDMDKeepIn* item provides more information about the keep-in (such as its purpose), and in turn references an *EDMDShapeElement* describing the keep-in's shape.

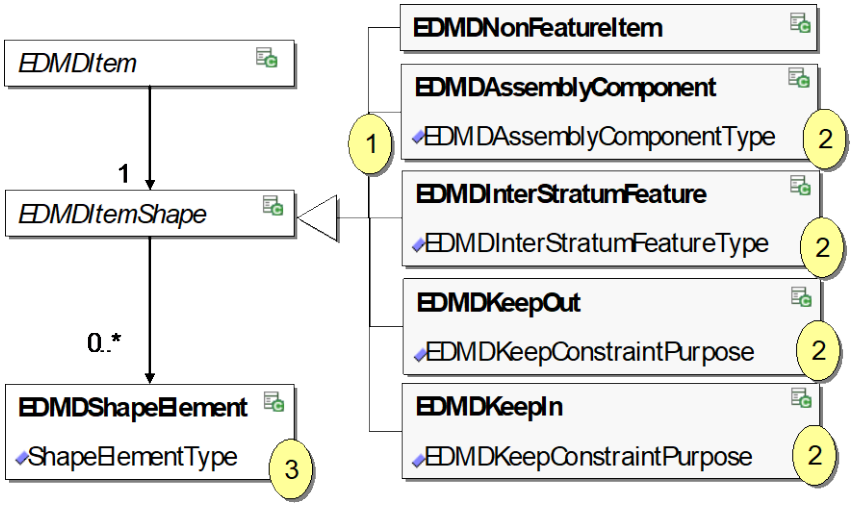


Figure 14: Three "parameters" for classifying an item using the 'original' method

The available combinations are listed below in Table 2. The column IDX Type shows schema attribute type and its values. For detailed description of the value see HTML documentation of the IDX schema (see PSI5_EDMD-Schema_4.0_HTML.zip).

IDX schema name (Parameter #1)	Description	**IDX** Type (Parameter #2)
EDMDNonFeatureItemShape	*EDMDNonFeatureItemShape* is an *EDMEItemShape*, that defines a non-feature shape which is related to other *EDMDItems*	***EDMDNonFeatureItemShapePurpose*** InfoText, InfoGeometry, AuxiliaryGeometry
EDMDAssemblyComponent	Assembly component	***EDMDAssemblyComponentType*** AssemblyGroupComponent, LaminateComponent, ThermalComponent, Physical, Padstack
EDMDFunctionalItemShape	*EDMDFunctionalItemShape* is an ItemShape, which has a special function. Function is described by ShapeElements.	***EDMDFunctionalItemShapeType*** N.N.
EDMDInterStratumFeature	*EDMDInterStratumFeature* spans one or more *EDMDStratum*.	***EDMDInterStratumFeatureType*** ***UnsupportedPassage, StratumFeatureTemplateComponent, CutoutEdgeSegment, PlatedCutoutEdgeSegment, Cutout, PlatedInterStratumFeature, PlatedPassage, PlatedCutout, Via, PhysicalConnectivityInterruptingCutout, PartiallyPlatedCutout, ComponentTerminationPassage, FilledVia, MechanicalComponent, CustomElectroMechanicalComponent, BoardCutout***
EDMDKeepOut	*EDMDKeepOut* is an *EDMDItemShape*, which defines a keep out constraint for a special purpose.	***EDMDKeepConstraintPurpose*** ***Thermal, GenericClearance, Shock, Vibration, ElectromagneticCompatibility, ComponentPlacement, Route, Via, TestPoint, ValueLimit***
EDMDKeepIn	EDMDKeepIn is an EDMDItemShape, which defines a keep in constrained for a special purpose.	***EDMDKeepConstraintPurpose*** ***Thermal, GenericClearance, Shock, Vibration, ElectromagneticCompatibility, ComponentPlacement, Route, Via, TestPoint, ValueLimit***
EDMDShapeElement	A shape_element is a portion of shape that has to be identified explicitly to be associated with other information.	***EDMDShapeElementType*** ***NonFeatureShapeElement, FeatureShapeElement, PartFeature, PartMountingFeature, ComponentTerminationPassageTemplateTerminal,***

		ComponentTerminationPassageTemplateInterfaceTerminal, ComponentTerminationPassageTemplateJoinTerminal, PrintedPartCrossSectionTemplateTerminal, PrintedPartTemplateInterfaceTerminal, PrintedPartTemplateJoinTerminal, StructuredPrintedPartTemplateTerminal, PrintedPartTemplateTerminal, PartToolingFeature, FiducialPartFeature, ThermalFeature, TestPointPartFeature, ConnectionZone, ComponentTerminal, BareDieComponentTerminal, Land, Package, PackageTerminal, ViaTerminal

Table 2: IDX ItemShape classes and their detailed types

Possible shape element types (Parameter #3) are for example *FeatureShapeElement*, *PartFeature*, *PartMountingFeature*, *ViaTerminal*, *PrintedPartTemplateInterfaceTerminal* etc. The complete list as well as description can be found in the HTML documentation.

4.1.2.2 How PCB features are described using the ‘simplified‘ method

By introducing a ‘geometryType‘ attribute, we can remove the need for specific objects to represent specific PCB features, such as the *EDMDKeepIn* object to describe a keep-in. As we have indicated the Item is a keep-in using the attribute we can omit this item and hence reduce the complexity of the XML required to describe each item. The specific examples in Section 6 show how to describe each feature using both methods.

The usage of the attribute values for classifying an item using the 'original' method is shown below:

Elements	(Parameter #1) EDMDShape schema name	(Parameter #2) EDMD Type (attribute name depends on concrete EDMDShape)	(Parameter #3) EDMD ShapeElementType	(Parameter #4)
Cutout	InterStratumFeature	Cutout	PartMountingFeature	
Mounting hole – non-plated	InterStratumFeature	Cutout	DesignLayerStratum	
Mounting hole – plated	InterStratumFeature	PlatedCutout	DesignLayerStratum	
Keep-in – Route Border	KeepIn	Route	FeatureShapeElement	
Keep-in – Route Border	KeepIn	Route	FeatureShapeElement	
Keep-out – Placement obstruct	KeepOut	ComponentPlacement	FeatureShapeElement	
Keep-out – plane keepout	KeepOut	Plane	FeatureShapeElement	
Keep-out – Route keepout	KeepOut	Route	FeatureShapeElement	
Keep-out – Via keepout	KeepOut	Via	FeatureShapeElement	
PCB component	AssemblyComponent	Physical	FeatureShapeElement	
PCB Mech Component	FunctionalItemShape	MechanicalItem	FeatureShapeElement	
Flex area	FunctionalItemShape	FlexibleArea	FeatureShapeElement	
Room	FunctionalItemShape	PlacementGroupArea	FeatureShapeElement	
User Defined Area	FunctionalItemShape	UserArea	FeatureShapeElement	
Copper pad	Stratum	DesignLayerStratum		**StratumTechnology LayerPurpose – LandsOnly**
Copper Trace	Stratum	DesignLayerStratum		**StratumTechnology LayerPurpose – OtherSignal**
Via	InterStratumFeature	DesignLayerStratum	PartMountingFeature	**InterStratumFeatureType - Via**
Bend Area	**Bend**			

Table 3: Values for "classification parameters" of an item supported by current implementations

4.1.3 Marking Items as Non-collaborative

In the IDX format the assumption is every item can be collaborated on, hence both MCAD and ECAD have the rights to add/remove/modify or move any item. However, in some cases one system may wish to specify that a certain item is non-collaborative and can only be modified by its owner.

In this case ideally the receiving system should prevent the modification of the item. If the Item is modified, any changes made would be automatically rejected.

Items are marked as non-collaborative at the Instance level. Subsequently instance C1 of a capacitor could be marked non-collaborative, but instance C2 of the same component could be freely moved. Modification of the component shape would not be permitted in this case however, as that would result in the modification of both instances including the non-collaborative instance C1.

For example, the XML below shows the top-level *EDMDItem* for a Component, where the owner is set as 'Mechanical', implying that it should not be modified by the Electrical system:

```
<foundation:Item id="ITEM210" IsAttributeChanged="false”>
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>BOARD ASSEMBLY</foundation:Description>
      <pdm:ItemType>assembly</pdm:ItemType>
       <!-- pdm:Identifier object containing item's ID -->
      <pdm:ItemInstance id="ITEMINST006" xsi:type="foundation:EDMDItemInstance">
           <pdm:Item>CMPITEM209</pdm:Item>
           <pdm:InstanceName IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
           <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
           <foundation:ObjectName>cc1210_pn-cc1210-1</foundation:ObjectName>
           </pdm:InstanceName>
           <foundation:Name>C1</foundation:Name>
          <!—Transformation can be 2D or 3D. -->
          <pdm:Transformation>
               <pdm:TransformationType>d2</pdm:TransformationType>
               <pdm:xx>0.62347</pdm:xx>
               <pdm:xy>-0.781847</pdm:xy>
               <pdm:yx>0.781847</pdm:yx>
               <pdm:yy>0.62347</pdm:yy>
               <pdm:tx IsAttributeChanged="false">
                   <property:Value>36.13551</property:Value>
            </pdm:tx>
            <pdm:ty IsAttributeChanged="false">
                 <property:Value>20.598426</property:Value>
            </pdm:ty>
        </pdm:Transformation>
     </pdm:ItemInstance>
</foundation:Item>
```

```
<!—RoleOnItemInstance object specifies ITEMINST006 above belongs to Mechanical
and should not be modified by Electrical -→

<foundation:RoleOnItemInstance>
      <administration:RoleName>
            <foundation:SystemScope>SYSTEM_1</foundation:SystemScope>
            <foundation:ObjectName>Owner</foundation:ObjectName>
       </administration:RoleName>
       <administration:RoleType>owner</administration:RoleType>
       <administration:Category>Mechanical</administration:Category>
       <administration:Function>Design</administration:Function>
       <administration:Context>ITEMINST006</administration:Context>
</foundation:RoleOnItemInstance>
```

5 IDX Message Types

5.1 The “SendInformation” message

The “SendInformation” message is used to send an initial design Baseline from ECAD to MCAD or vice-versa. This Baseline represents the entire extent of the design to be collaborated on at that point in time. Once ECAD or MCAD has received and processed the initial Baseline it is assumed each user has the same synchronized version of the PCB data.

“SendInformation” messages can also be used to reset the collaboration state by ‘re-baselining’, or to send additional non-collaborative data such as traces. See section 3.1.2 for more information.

A *SendInformation* XML document consists of one *EDMDDataSet* object, which contains three sections (see Figure 15):

- **Header**: Meta Information about the creator of the XML document (E.g. Company, Postprocessor etc.)
- **Body**: Specification of the items (*EDMDItems*) of the board (e.g. PCB board, cutouts, keep-outs, mounting holes, PCB components). A description of how to model features in the Body can be found in Sections 6 & 7).
- **ProcessInstructions**: A single instruction to indicate this message is of type *EDMDProcessInstructionSendInformation*.

```
<foundation:EDMDDataSet>

  <foundation:Header xsi:type="foundation:EDMDHeader">

    <!-- Header Information -->

  </foundation:Header>

  <foundation:Body xsi:type="foundation:EDMDDataSetBody">

    <!-- Geometrical Information -->

  </foundation:Body>

  <foundation:ProcessInstruction xsi:type="computational:EDMDProcessInstructionSendInformation"/>

</foundation:EDMDDataSet>
```

Figure 15: XML representation of SendInformation message (root)

Note: An *EDMDItem* describes a defined state of an element, not the element itself. The assignment of the element states to elements in the CAD system takes place in the CAD system via the CAD-system-specific designator (see *EDMDDesignator*) for the *EDMDItem*. As a rule, the complete description of an element is not transported in a message but rather the differences between two states of an element.

5.2 The “SendChanges” message

5.2.1 Sending Proposed Changes using “SendChanges”

The “SendChanges” message is used to send both proposed changes and responses to those changes from ECAD to MCAD or vice-versa.

- Files or messages sending proposed changes are known as “Change Files”.
- Files or messages responding to proposed changes in Change Files as known as “Response Files”.

This subsection looks at the structure of “Change Files”; the next section looks at “Response Files”.

Unlike a “SendInformation” message which represents the entire extent of the design, “SendChanges” messages only contain information about the item or items to be changed.

One change message may consist of several proposed changes. The collaboration partner may accept some and reject the other. In order to support the functionality by ECAD or MCAD systems, each item instance is related with an *EDMDItem* (of type assembly). This item is used to express changes on the related instance (item instance).
Figure 16 illustrates the representation of a change based on a movement of instance A1_inst of PCB Package A.

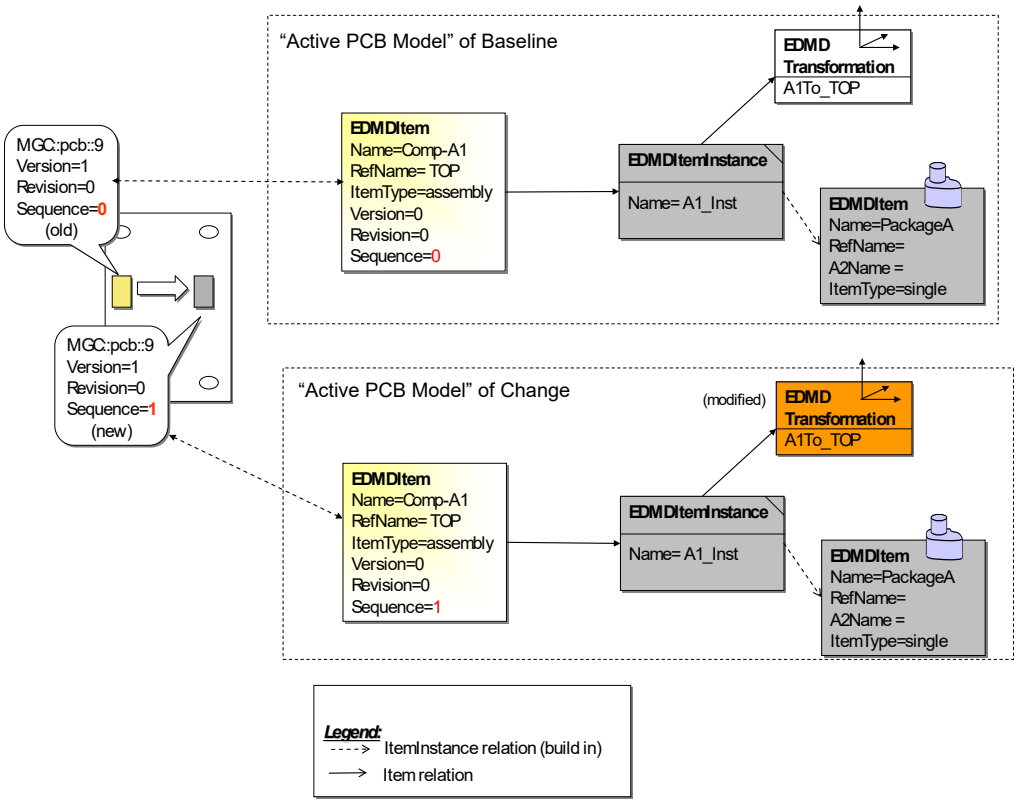


Figure 16: Representation of a proposed change (here: Movement of a component)

An XML document for representing a SendChanges message is structured in the same basic form as the SendInformation message used to send the initial design baseline. It consists of:

- A parent *EDMDDataSet* object which contains:
- *Header*: Information about the file type, creator etc.

- *Body*: Describes the geometry and position of the new items.
- *ProcessInstruction* object of type *EDMDProcessInstructionSendChanges* which contains:
  - *Changes*: A list of proposed changes which reference items in the Body (see use case 3).

A typical “SendChanges” message would look like this *(some XML removed for clarity):*

```
<foundation:EDMDDataSet>
        <foundation:Header>
                <!--Header information -->
        </foundation:Header>
        <foundation:Body>
                <!--Geometric information -->
        </foundation:Body>
        <foundation:ProcessInstruction xsi:type="computational:EDMDProcessInstructionSendChanges">
                <!--Change information -->
        </foundation:ProcessInstruction>
</foundation:EDMDDataSet>
```

Figure 17: XML representation of SendChange message

The *ProcessInstructions* object contains one or more proposed changes to the board. Where changes reference geometry this will be contained within the *Body* object and described in the same way as within the *Body* object in a “SendInformation” file.

Each single change specifies both the new item and the predecessor item by its *EDMDIdentifier*. Each *EDMDIdentifier* contains a tuple or identifier (a unique combination of *SystemScope, Number, Version, Revision* & *Sequence*) which identifies that item. The *Sequence* attribute is incremented to indicate a change to an existing item.

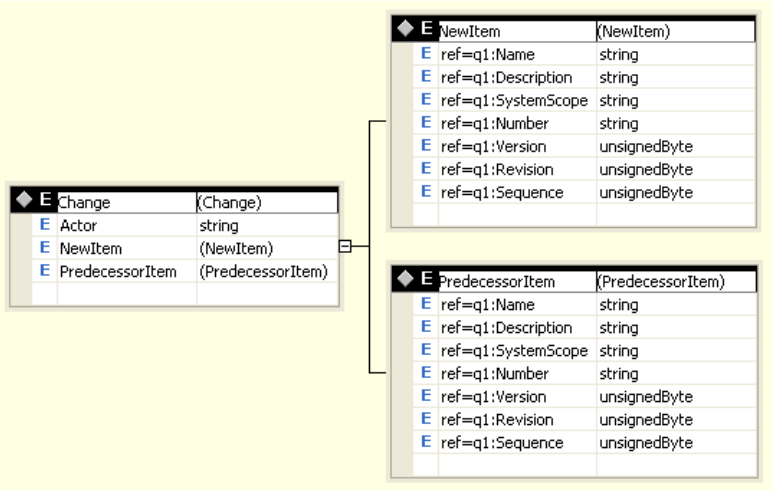


Figure 18: Definition of a single proposed change

Multiple Changes can be sent in one ProcessInstruction in the format shown below *(some XML removed for clarity):*

```
<foundation:EDMDDataSet>
    <foundation:Header>
        <!—Header Information  -->
    </foundation:Header>
    <foundation:Body>
        <!--Geometric information -->
    </foundation:Body>
    <foundation:ProcessInstruction xsi:type="computational:EDMDProcessInstructionSendChanges">
        <computational:Actor>rlacey</computational:Actor>
        <computational:Changes xsi:type="computational:EDMDTransaction">
            <!—First Proposed Change -->
        </computational:Changes>
        <computational:Changes xsi:type="computational:EDMDTransaction">
            <!—Second Proposed Change -->
        </computational:Changes>
        <computational:Changes xsi:type="computational:EDMDTransaction">
            <!—Third Proposed Change -->
        </computational:Changes>
</foundation:EDMDDataSet>
```

Figure 19: Overview of SendChanges message in XML

The following section describes the “Changes” structure based on an example.

The XML shown in Figure 20 describes the change of an Item. The user that initiated the change is specified in the tag *<Actor>.*

Item A (the *PredecessorItem*) has the identifier:

- Number: *MGC::ExpeditionPCB_candy26::4*
- SystemScope: *ECADSYSTEM*
- Version: *1*
- Revision: *0*
- Sequence: *0*

And is to be exchanged for Item B (the *NewItem*):

- Number: *MGC::ExpeditionPCB_candy26::4*
- SystemScope: *ECADSYSTEM*
- Version: *1*
- Revision: *0*
- Sequence: *1*

```
<foundation:ProcessInstruction id="MGC::46::12" xsi:type="computational:EDMDProcessInstructionSendChanges">
  <computational:Actor>rlacey</computational:Actor>
  <computational:Changes xsi:type="computational:EDMDTransaction">
    <foundation:Description></foundation:Description>
    <computational:Change xsi:type="computational:EDMDChange">
      <computational:Actor>rlacey</computational:Actor>
      <computational:NewItem xsi:type="foundation:EDMDIdentifier">
        <foundation:Name>PCB</foundation:Name>
        <foundation:Description>THIS IS THE CHANGED VERSION</foundation:Description>
        <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
        <foundation:Number>MGC::ExpeditionPCB_candy26::4</foundation:Number>
        <foundation:Version>1</foundation:Version>
        <foundation:Revision>0</foundation:Revision>
        <foundation:Sequence>1</foundation:Sequence>
      </computational:NewItem>                                   NEW ITEM (Sequence 1)
      <computational:PredecessorItem xsi:type="foundation:EDMDIdentifier">
        <foundation:Name>PCB</foundation:Name>
        <foundation:Description>THIS IS THE PREVIOUS VERSION</foundation:Description>
        <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
        <foundation:Number></foundation:Number>
        <foundation:Version>1</foundation:Version>
        <foundation:Revision>0</foundation:Revision>
        <foundation:Sequence>0</foundation:Sequence>
      </computational:PredecessorItem>                           OLD ITEM (Sequence 0)
    </computational:Change>
  </computational:Changes>
</foundation:ProcessInstruction>
```

Figure 20: Example for instantiation of the "Changes" section

Both items are described in the Body (*EDMDDataSetBody*) section of the document. The link between the item in “Changes” and “Dataset” section is established by the *EDMDIdentifier* (see Figure 21).

Note that the *NewItem* in the Change has the identifier:

- Number: *MGC::ExpeditionPCB_candy26::4*
- SystemScope: *ECADSYSTEM*
- Version: *1*
- Revision: *0*
- Sequence: *1*

As shown below, the same identifier can also be found in the main *Body/EDMDDataSet* section of the file.

The shape and position of the item is also described in the *EDMDDataset* in exactly the same way as it would be in a Baseline or “SendInformation” message. See sections 6 and 7 for information about how to describe the shape and position of an item/instance.

```
<foundation:Header xsi:type="foundation:EDMDHeader">

<foundation:Body xsi:type="foundation:EDMDDataSetBody">
  <foundation:Item id="ITEM46" xsi:type="pdm:EDMDItem">
    <foundation:Name>PCB</foundation:Name>
    <foundation:Description>NEW BOARD DESCRIPTION</foundation:Description>
    <pdm:ItemType>assembly</pdm:ItemType>
    <pdm:Identifier xsi:type="foundation:EDMDIdentifier">
      <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
      <foundation:Number>MGC::ExpeditionPCB_candy26::4</foundation:Number>
      <foundation:Version>1</foundation:Version>
      <foundation:Revision>0</foundation:Revision>
      <foundation:Sequence>1</foundation:Sequence>
    </pdm:Identifier>
    <pdm:ItemInstance id="ITEMINSTANCE5" xsi:type="pdm:EDMDItemInstance">
    <pdm:AssembleToName>INNER_Layer_2</pdm:AssembleToName>
  </foundation:Item>
  <!--- Other Geometry Items --->
</foundation:Body>

<foundation:ProcessInstruction id="MGC::46::12" xsi:type="computational:EDMDProcessInstructionSendChanges">
  <computational:Actor>rlacey</computational:Actor>
  <computational:Changes xsi:type="computational:EDMDTransaction">
    <foundation:Description></foundation:Description>
    <computational:Change xsi:type="computational:EDMDChange">
      <computational:Actor>rlacey</computational:Actor>
      <computational:NewItem xsi:type="foundation:EDMDIdentifier">
        <foundation:Name>PCB</foundation:Name>
        <foundation:Description>THIS IS THE CHANGED VERSION</foundation:Description>
        <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
        <foundation:Number>MGC::ExpeditionPCB_candy26::4</foundation:Number>
        <foundation:Version>1</foundation:Version>
        <foundation:Revision>0</foundation:Revision>
        <foundation:Sequence>1</foundation:Sequence>
      </computational:NewItem>
      <computational:PredecessorItem xsi:type="foundation:EDMDIdentifier">
    </computational:Change>
  </computational:Changes>
</foundation:ProcessInstruction>
```

Figure 21: Description of items in “change” section is done in “Dataset” section

Note that the SystemScope attribute should be set by the system that initially created the feature and should subsequently remain unchanged throughout subsequent changes. The SystemScope attribute’s purpose is to act as a unique namespace within the collaboration to ensure there are no clashes between the identifiers used by each system. In the example above the SystemScope of item number “MGC::pcb::9” would remain as “ECADSYSTEM” even if a Change file was written by MCAD. In that case the Actor would be defined as “MCAD” rather than “ECAD”, but the SystemScope would still be “ECADSYSTEM”.

There are several types of change supported by the IDX schema:

- **Add:** E.g. add a new item (instance) which did not exist in the previous version of the PCB. In this situation a new item instance (e.g. a component) is added and the containing item assembly (e.g. PCB board) is modified. The new item is specified in the *NewItem* section of the Change, and no PredecessorItem is specified. The geometry of the new item is described within the *EDMDDataSetBody* section.
- **Remove**: Remove instances of an item (e.g. component). In this situation the item assembly (e.g. PCB board) will be modified. The item to be deleted is referenced in *NewItem* in the same format as when adding a Component but with no corresponding Item in defined in *EDMDDataSetBody*. Additionally, the object *DeletedInstanceName* added to the *Change* object as shown below in Figure 22 to indicate a deletion rather than an addition:

- **Change:** Changes in attributes of item version: The new item version needs to be marked as *NewItem* in the *Change* section. The optional attribute *IsAttributeChanged* can be used to indicate that some attributes of this instance are changed in this collaboration object. This attribute is set in the section which describes the properties of the item. If this attribute is not set the message interpreting system has to recognize if attributes are changed by calculation differences to predecessor.

```
<foundation:ProcessInstruction xsi:type="computational:EDMDProcessInstructionSendChanges" id="SW::test_idf::07/12/19 12:50:49">
  <computational:Actor>MCAD</computational:Actor>
  <computational:Changes xsi:type="computational:EDMDTransaction" id="TRANSACTION1">
    <foundation:Name>rlacey::7/12/2019 12:50:49 PM</foundation:Name>
    <foundation:Description>Removed J2 note</foundation:Description>
    <computational:Actor>MCAD</computational:Actor>
    <computational:Change id="SWX_636985326495117311" IsAttributeChanged="false">
      <computational:Actor>MCAD</computational:Actor>
      <computational:NewItem IsAttributeChanged="false" Persistant="false" IsOriginator="false">
        <foundation:Description>THIS IS THE CHANGED VERSION</foundation:Description>
        <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
        <foundation:ObjectName>PCB</foundation:ObjectName>
        <foundation:Number>conn_din24_connector-2</foundation:Number>
        <foundation:Version>1</foundation:Version>
        <foundation:Revision>0</foundation:Revision>
        <foundation:Sequence>1</foundation:Sequence>
      </computational:NewItem>
      <computational:DeletedInstanceName IsAttributeChanged="false" Persistant="false" IsOriginator="false">
        <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
        <foundation:ObjectName>conn_din24_connector-2</foundation:ObjectName>
      </computational:DeletedInstanceName>
    </computational:Change>
  </computational:Changes>
</foundation:ProcessInstruction>
```

Figure 22: Addition of ‘DeletedInstanceName’ object to indicate item to be deleted

5.2.2 Responding to Proposed Changes using “SendChanges”

The “SendChanges” message format is also used to reply to a “SendChanges” message containing proposed changes (see Section 3 for information about IDX workflow).

Like a “SendChanges” message proposing a change (a “Change file”) described in the preceding section, the “SendChanges” message used to respond (a “Response file”) only contains information about the item or items to be changed.

A typical “Response file” is shown below:

```
<foundation:Header xsi:type="foundation:EDMDHeader">

<foundation:Body xsi:type="foundation:EDMDDataSetBody">
  <foundation:Item id="ITEM46" xsi:type="pdm:EDMDItem">
    <foundation:Name>PCB</foundation:Name>
    <foundation:Description>NEW BOARD DESCRIPTION</foundation:Description>
    <pdm:ItemType>assembly</pdm:ItemType>
    <pdm:Identifier xsi:type="foundation:EDMDIdentifier">
      <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
      <foundation:Number>MGC::ExpeditionPCB_candy26::4</foundation:Number>
      <foundation:Version>1</foundation:Version>
      <foundation:Revision>0</foundation:Revision>
      <foundation:Sequence>1</foundation:Sequence>
    </pdm:Identifier>
    <pdm:ItemInstance id="ITEMINSTANCE5" xsi:type="pdm:EDMDItemInstance">
      <pdm:Accept>
        <property:Value>false</property:Value>
      </pdm:Accept>
      <pdm:AssembleToName>INNER_Layer_2</pdm:AssembleToName>
    </foundation:Item>
    <!--- Other Geometry Items --->
</foundation:Body>

<foundation:ProcessInstruction id="MGC::46::12" xsi:type="computational:EDMDProcessInstructionSendChanges">
  <computational:Actor>rlacey</computational:Actor>
  <computational:Changes xsi:type="computational:EDMDTransaction">
    <foundation:Description>Not a good location</foundation:Description>
    <computational:Change xsi:type="computational:EDMDChange">
      <computational:Actor>rlacey</computational:Actor>
      <computational:NewItem xsi:type="foundation:EDMDIdentifier">
        <foundation:Name>PCB</foundation:Name>
        <foundation:Description>THIS IS THE CHANGED VERSION</foundation:Description>
        <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
        <foundation:Number>MGC::ExpeditionPCB_candy26::4</foundation:Number>
        <foundation:Version>1</foundation:Version>
        <foundation:Revision>0</foundation:Revision>
        <foundation:Sequence>1</foundation:Sequence>
      </computational:NewItem>
      <computational:PredecessorItem xsi:type="foundation:EDMDIdentifier">
    </computational:Change>
  </computational:Changes>
</foundation:ProcessInstruction>
```

Figure 23: Addition of ‘Accept’ object in ‘Response’ file

The structure of a Response File is virtually identical to the structure of the Change file it is responding to. The file contains the same Header, Body, and ProcessInstruction sections (still of type *EDMDProcessInstructionSendChanges*) as the Change file. However, the items referenced by the Changes in the *EDMDDataSetBody* utilize the Boolean *<pdm.Accept>* attribute to indicate if the user wishes to accept or reject that proposed change, shown here in green.

In the example in the preceding section, a change to the item with this identifier was proposed:

- Number: *MGC::ExpeditionPCB_candy26::4*
- SystemScope: *ECADSYSTEM*
- Version: *1*
- Revision: *0*
- Sequence: *1*

If the receiving system chose to reject that change, the corresponding “Response file” would be structured as shown above. Note that it is virtually identical to the “Change file” apart from the <*pdm:Accept*> attribute highlighted in green.

The user can also add a comment saying why they have accepted or rejected the change(s) utilizing the *Description* attribute within the *Changes* object. In the example above the user has added the comment *“Not a good location”*.

To accept a proposed change *true* would be used for the value of <*pdm:Accept*>. Comments can also be returned if required confirming the reason the change was accepted.

5.3 Timestamps in IDX messages

IDX files need to maintain consistent timestamp information to ensure that files can be processed in the correct order. As IDX communication can span several time zones it is essential that timestamps are expressed consistently specifying their time zone where appropriate. This will ensure that systems processing IDX files can still correctly process files in timestamp order even if they were created in different time zones.

Version 4.0 of the IDX standard defines two timestamps within the *EDMDDataSet* object.

The first, *CreationDateTime* represents the original date and time when the dataset was initially created. A second timestamp *ModifiedDateTime*, specifies the last time the dataset was modified.

Example:

```
<foundation:EDMDDataSet>
  <foundation:Header xsi:type="foundation:EDMDHeader">
    <foundation:Description>Test PCB design</foundation:Description>
    <foundation:CreatorName></foundation:CreatorName>
    <foundation:CreatorCompany>MGC</foundation:CreatorCompany>
    <foundation:CreatorSystem>EE2007</foundation:CreatorSystem>
    <foundation:PostProcessor></foundation:PostProcessor>
    <foundation:PostProcessorVersion>2008.x.xxx.xxx</foundation:PostProcessorVersion>
    <foundation:Creator>shrogers</foundation:Creator>
  <foundation:GlobalUnitLength>UNIT_MM</foundation:GlobalUnitLength>
  <foundation:CreationDateTime>2012-08-29T10:28:27Z</foundation:CreationDateTime>
  <foundation:ModifiedDateTime>2012-08-29T10:28:27Z</foundation:ModifiedDateTime>
  </foundation:Header>
  <foundation:Body>…</foundation:Body>
  <foundation:ProcessInstruction>…</foundation:ProcessInstruction>
  <foundation:ProcessInstruction>…</foundation:ProcessInstruction>
</foundation:EDMDDataSet>
```

The element name *CreationDateTime* was selected to be consistent with other elements in the IDX schema that record the date and time (i.e. *EDMDHistoryEntry*). In the schema file the *CreationDateTime* element would be defined with the following:

```
<xs:element name="CreationDateTime" type="xs:dateTime">
      <xs:annotation>
            <xs:documentation>
                  Creation date and time of this IDX proposal. Responses should use the
                  same timestamp as the proposal as the timestamp is used to link the
                  response to the appropriate proposal.
            </xs:documentation>
      </xs:annotation>
</xs:element>
```

The ModifiedDateTime element indicates when the last time the IDX file was modified, often to add a response to the proposal.

```
<xs:element name="ModifiedDateTime" type="xs:dateTime">
```

```
    <xs:annotation>
        <xs:documentation>
            Date and time when this IDX file was modified, often to include a response.

        </xs:documentation>
    </xs:annotation>
</xs:element>
```

Using a consistent *CreationDateTime* timestamp throughout all the IDX files related to a specific design ensures that the correct IDX files are applied to the correct design. The same folder could contain two sets of files relating to a board called 'PCB' for example, but as each set would have a different *CreationDateTime* timestamp, the processing system would be able to differentiate between files belonging to each data set. The progression of the *ModifiedDateTime* timestamp would determine the order files in each data set should be processed.

The dateTime data type is a predefined type in XML Schema that supports time zones. For more details about the dateTime type see *http://www.w3schools.com/schema/schema_dtypes_date.asp*

5.4 Transaction History in IDX messages

NOTE: This is optional. Transaction history is not currently mandatory and is only supported by some implementations of IDX. The History objects are not currently part of the formal IDX schema.

When a change is proposed, accepted, or rejected the user should add a comment that provides some context. Ideally this comment must have information to identify who created it, when, from what tool etc.

Representing Transaction History in IDX:

- Associate history with a transaction or change

  History is a collection of History entries. Each History entry collection must then be associated with a transaction or change that it applies to. This can be accomplished by adding a Subject element (reference to *EDMDObject*) to *EDMDHistory*. This will allow association of a group of history entries with an object (e.g. *EDMDTransaction*)

- Set the Application that Created a History Entry

  Add SystemScope (reference to *EDMDSystem*) to *EDMDHistoryEntry*.

- Set the Application Revision

  Add Revision to *EDMDSystem*.

Example:

The following XML example shows how to represent Transaction History in IDX:

```
<History id=HISTORY_1>
     <!—Added for 2.3.1 Associate history with a change -->
     <Subject>CHANGE_1</Subject>
</History>

<HistoryEntryChangeAcceptance id=HISTORYENTRY_1>
     <Subject>ACCEPT_1</Subject>
     <Creator>USER_1</Creator>
     <Text>I don’t like this change.</Text>
     <!—Added for 2.3.2 Set the application that created a history entry -->
     <SystemScope>PTC_CREO_PMA</SystemScope>
     <History>HISTORY_1</History/>
     <CreationDateTime>October 16, 2012 12:00 GMT</CreationDateTime>
</HistoryEntryChangeAcceptance>

<!—Referenced by the HISTORYENTRY_1 to set the user information -->
<Person id=USER_1>
     <SurName>Steve</SurName>
     <GivenName>Smith</GivenName>
```

```
      <EMailAddress>user@example.com</EMailAddress>
      <PhoneNumber>111-222-333</PhoneNumber>
</Person>

<!—Referenced by the HISTORYENTRY_1 to set the tool name -->
<System id=PTC_CREO_PMA>
       <SystemType>MCAD</SystemType>
       <GlobalName>Creo Parametric</GlobalName>
       <!—Added for 2.3.3 Set the Application Revision -->
       <Revision>1.0</Revision>
</System>

<Item id=ITEM_1>
       <ItemType>Assembly</ItemType>
       <Identifier>...</Identifier>
       <ItemInstance>...</ItemInstance>
<!—Referenced by the history entry to indicate acceptance state -->
       <Accept id=ACCEPT_1>
       <Value>false</Value>
       </Accept>
</Item>

<!—Referenced by HISTORY_1 to associated with a change -->
<ProcessInstruction>
       <Changes id=TRANSACTION_1 xsi:type=EDMDTransaction>
              <Change id=CHANGE_1>
              <NewItem>...</NewItem>
              <PredecessorItem>...</PredecessorItem>
              </Change>
       </Changes>
</ProcessInstruction>
```

5.5 Sending IDX Messages as Files

5.5.1 File Suffix

When files are used to transfer IDX messages, the default file suffix to be used for standard uncompressed IDX files is **.idx*. The same suffix is used for all file types (Baseline, Change and Response files).

5.5.2 Use of message file compression (*.idz)

Due to the nature of XML and the complexity of the IDX format, some messages (particularly the Baseline message), can get very large when saved as files. Therefore, one message file may be put in zip file. In order to indicate that this file is a compressed idx file the extension **.idz* should be used rather than the **.idx* suffix used for uncompressed IDX data. From a user perspective the handling may the same as with **.idx* files.

Any compression/uncompression should be executed by the ECAD/MCAD system after detecting the *.idz* extension.

The recommended compression format should be the Huffman-based 'Deflate' algorithm as used by standard compression programs such as Zip, PKZIP and the PNG image format.

For more information see: https://en.wikipedia.org/wiki/DEFLATE

7zip (see: http://www.7-zip.org) must be able to process the file.

5.5.3 File Naming and Folder Usage

If IDX messages are transferred by *.idx* or *.idz* files it's recommended a clear naming structure is used to differentiate between IDX file types (e.g. Baselines, Changes, Responses etc.). For example, files relating to PCB design called "Cellphone" could be named:

Cellphone_baseline_00.idx
Cellphone_change_01.idx
Cellphone_response_01.idx
Cellphone_change_02.idx etc.

The software processing the files should user Header and Timestamp information to determine the file type and order rather than filenames but using logical file naming makes life easier for users.

The use of subfolders is also recommended in order to keep the files relating to each PCB design together and separate from other designs in order to avoid systems inadvertently processing changes or responses intended for another design (although correct processing of each IDX file should avoid this).

6 How to model specific ECAD features in IDX

This section describes in more detail how to define typical ECAD items in the IDX format. Note that for brevity the XML describing the shape(s) of each Item is generally omitted from the examples. Please see the following section for information on how to describe 2D and 3D shapes.

Versions of the IDX schema prior to IDXv4.0 used the same basic hierarchy of objects to describe virtually every item on the PCB (BOARD, HOLE, KEEPOUT, COMPONENT etc.)

The description of each object is very similar for each item and follows the same basic pattern of:

```
Item (assembly) >
      Item (single) >
            "Third Item" (EDMDStratum, EDMDKeepin etc.) >
                  ShapeElement >
                        CurveSet2d >
                              Polyline >
                                    Points
```

The type of the “Third Item” tells us what the object is (e.g. is it a *Keepout*, a *Component* etc.), and also provides more information about the object such as its specific type (e.g. Routing Keepout).

There are a couple of disadvantages to this method of defining ECAD items: Firstly, it’s very long, so a lot of XML is needed to define each item, and secondly it’s difficult for programs reading an IDX file to work out what an object represents until the “Third Item” is processed.

To help address both these issues, IDXv4.0 introduced the ‘geometryType’ attribute. This is an attribute on the first Item (assembly) which specifies what the item and its child items represent (e.g. *keepout*, *component* etc.). In most (but not all) cases, including this *geometryType* attribute means that the “Third Item” can be omitted, so the basic pattern becomes:

```
Item (assembly) geometryType = [type] >
       Item (single) >
            ShapeElement >
                  CurveSet2d >
                        Polyline >
                              Points
```

This reduces the amount of XML needed to define each item and makes the XML more readable both to humans and programs parsing the text.

IDXv4.0 supports both ways of describing an object; the “Traditional” method used prior to IDXv4.0 and the new “Simplified” method described above introduced in IDXv4.0. Although IDXv4.0 supports both methods and this section describes how to define each item using both methods, it’s strongly recommended that the new “Simplified” method is used for IDXv4.0 as the “Traditional” method will be depreciated in a future version of the IDX format.

A complete list of the possible *geometryType* attributes is given at the end of this section.

6.1 How to model the Board

The IDX format allows for several ways of defining the shape of the board, ranging from a simple outline and thickness similar to the IDF format, to more complex multi-layer, flexible and flex-rigid board descriptions.

This is the format to use to describe a simple board without any defined layers. This board has a single thickness of 1.5748mm and a cutout through the entire thickness of the board.

6.1.1 Describing a simple Board in IDX

1) The top level *EDMDItem* of type "assembly" represents the Board instance. The *geometryType="BOARD_OUTLINE"* attribute on the first line tells us this item and its children represent the Board. This attribute was added for IDXv4.0, and whilst not mandatory, if included means the *EDMDStratum* object can be omitted. See the previous section for more information on the new simplified format introduced in IDXv4.0.

```
<foundation:Item id="ITEM3" geometryType="BOARD_OUTLINE"
IsAttributeChanged="false">
     <foundation:Name>PCB</foundation:Name>
     <foundation:Description>BOARD ASSEMBLY</foundation:Description>
     <pdm:ItemType>assembly</pdm:ItemType>
       <!-- pdm:Identifier object containing item's ID -->
      <pdm:ItemInstance>
           <pdm:Item>BRDITEM2</pdm:Item>
           <!—User property specifying item is a Board Outline -->
           <foundation:UserProperty xsi:type="property:EDMDUserSimpleProperty"
IsChanged="false" IsNew="false">
                <property:Key IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
                     <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
                     <foundation:ObjectName>TYPE</foundation:ObjectName>
                </property:Key>
                <property:Value>BOARDOUTLINE</property:Value>
           </foundation:UserProperty>
           <!—Optional user properties -->
           <foundation:UserProperty xsi:type="property:EDMDUserSimpleProperty"
IsChanged="false" IsNew="false">
               <property:Key IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
                    <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
                    <foundation:ObjectName>THICKNESS</foundation:ObjectName>
               </property:Key>
               <property:Value>1.5748</property:Value>
          </foundation:UserProperty>
          <pdm:InstanceName IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
               <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
               <foundation:ObjectName>BoardOutline</foundation:ObjectName>
          </pdm:InstanceName>
     </pdm:ItemInstance>
</foundation:Item>
```

2)

EDMDItem of type single represents the Board:

```
<foundation:Item id="BRDITEM2" IsAttributeChanged="false">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>BOARD GEOMETRY</foundation:Description>
      <pdm:ItemType>single</pdm:ItemType>
      <!-- pdm:Identifier object containing item's ID -->
      <pdm:Shape>BOARDSTM1</pdm:Shape>
</foundation:Item>
```

3) The *EDMDItem*'s Shape references a Stratum object of *StratumType* "DesignLayerStratum" and *StratumSurfaceDesignation* of "PrimarySurface" which represents the board stratum/layer: In this case two *ShapeElements* are referenced by the Stratum, one which describes the board outline, and one which describes the cutout.

The IDXv4.0 version of the format allows for this object to be omitted providing the *geometryType="BOARD_OUTLINE"* attribute has been added to the top-level assembly item. In this case the *EDMDItem* (2) can reference the *ShapeElement* (4) below directly. However, the Stratum object must be included if the board outline has more than one shape, as is the case here.

```
<foundation:Stratum id="BOARDSTM1" IsAttributeChanged="false">
      <pdm:ShapeElement>BRDSHAPE1</pdm:ShapeElement>
      <pdm:ShapeElement>BRDSHAPE2</pdm:ShapeElement>
      <pdm:StratumType>DesignLayerStratum</pdm:StratumType>

<pdm:StratumSurfaceDesignation>PrimarySurface</pdm:StratumSurfaceDesignation>
</foundation:Stratum>
```

4) The Stratum in turn references two *ShapeElement* of type *FeatureShapeElement*. The first has Inverted set to false, indicating it is a boss or solid body rather than a cut. The second *ShapeElement* has Inverted set to true indicating a cut:

```
<foundation:ShapeElement id="BRDSHAPE1" IsAttributeChanged="false">
      <pdm:ShapeElementType>FeatureShapeElement</pdm:ShapeElementType>
      <pdm:DefiningShape>BRDCURVESET1</pdm:DefiningShape>
      <pdm:Inverted>false</pdm:Inverted>
</foundation:ShapeElement>

<foundation:ShapeElement id="BRDSHAPE2" IsAttributeChanged="false">
      <pdm:ShapeElementType>FeatureShapeElement</pdm:ShapeElementType>
      <pdm:DefiningShape>BRDCURVESET2</pdm:DefiningShape>
      <pdm:Inverted>true</pdm:Inverted>
</foundation:ShapeElement>
```

5) The *ShapeElements* in turn reference *CurveSet2D* objects describing the lines and arcs that make up both the closed board outer profile and the closed inner cut profile. BRDCURVESET1 (which represents the outer profile spans the range z=0 to z=1.5748 so is 1.5748mm high. BRDCURVESET2 (which represents the cutout) has a value of z=0 for both the Upper and Lower Bounds, this is assumed to be a 'Through all' cutout. A range of z=0 to z=1.5748 (the same thickness as the board) would also produce the same 'Through all' result:

```
<foundation:CurveSet2d xsi:type="d2:EDMDCurveSet2d" id="BRDCURVESET1"
IsAttributeChanged="false">
      <pdm:ShapeDescriptionType>GeometricModel</pdm:ShapeDescriptionType>
      <d2:LowerBound IsAttributeChanged="false">
            <property:Value>0</property:Value>
      </d2:LowerBound>
      <d2:UpperBound IsAttributeChanged="false">
            <property:Value>1.5748</property:Value>
      </d2:UpperBound>
```

```
<d2:DetailedGeometricModelElement>POLYLINE1</d2:DetailedGeometricModelElement>

<d2:DetailedGeometricModelElement>BRDARC1</d2:DetailedGeometricModelElement>

<d2:DetailedGeometricModelElement>POLYLINE2</d2:DetailedGeometricModelElement>

<d2:DetailedGeometricModelElement>BRDARC2</d2:DetailedGeometricModelElement>

<d2:DetailedGeometricModelElement>POLYLINE3</d2:DetailedGeometricModelElement>
</foundation:CurveSet2d>

<foundation:CurveSet2d xsi:type="d2:EDMDCurveSet2d" id="BRDCURVESET2"
IsAttributeChanged="false">
      <pdm:ShapeDescriptionType>GeometricModel</pdm:ShapeDescriptionType>
      <d2:LowerBound IsAttributeChanged="false">
            <property:Value>0</property:Value>
      </d2:LowerBound>
      <d2:UpperBound IsAttributeChanged="false">
            <property:Value>0</property:Value>
      </d2:UpperBound>

<d2:DetailedGeometricModelElement>BRDCIRCLECENTER1</d2:DetailedGeometricModelEle
ment>
</foundation:CurveSet2d>
```

6) Finally, the individual *DetailedGeometricModelElements* are defined, along with their child points. This XML is not shown for brevity.

6.1.2 Describing Boards in terms of their Layers

Modern Printed Circuit Boards are made up of alternating layers of conductive and insulating materials creating a ‘Layer Stackup’. Boards could be as simple as a layer of insulating dielectric material with a layer of conductive copper material on top or consist of many layers of alternative dielectric and signal layers, along with layers of soldermask, silkscreen or other features.

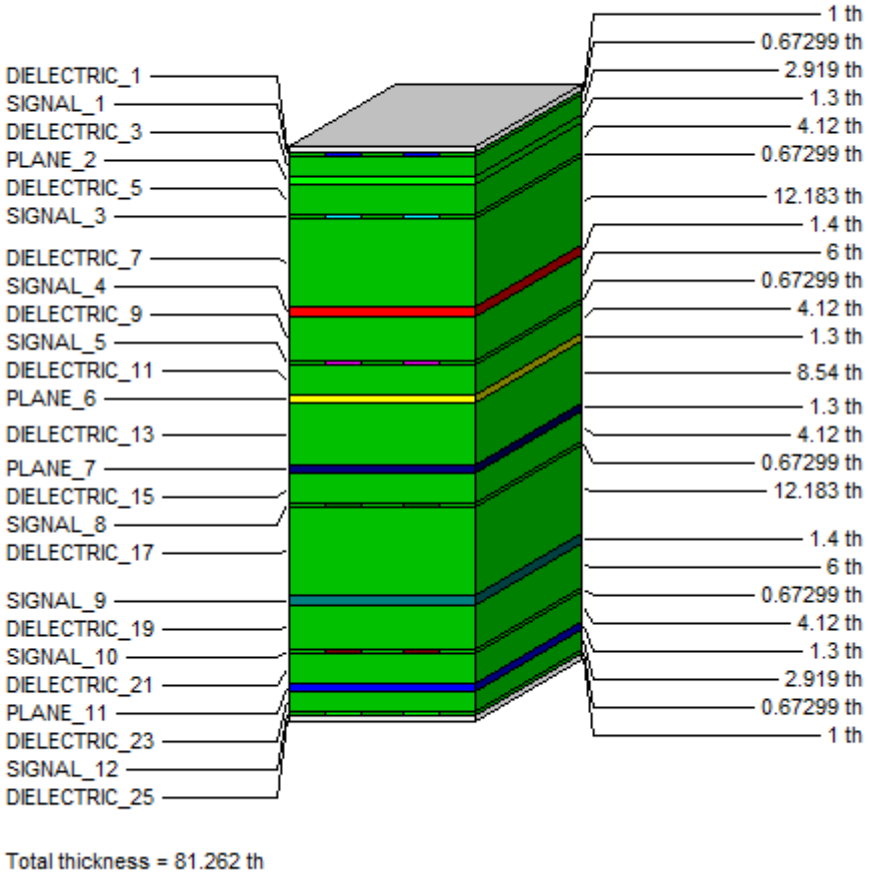


Figure 24: A cross-section of a typical PCB showing the layer structure or Layer “Stackup”

Whist these layers will always be represented in the ECAD system, it is not always necessary to represent them in the MCAD system. Often just giving the board a single ‘thickness’ as described above may suffice.

However, in some cases representation of board layers in the MCAD system may be required. Some boards may have different numbers of layers in different areas (and hence a different overall thickness). Components on different layers will subsequently be at different heights or could even be mounted within the board. For these more complex board designs, it’s necessary to define the board layers, how they’re stacked up (there may be more than one Stackup if different areas of the board have different number of layers), and the zones or areas of the board associated with a particular Stackup.

This model of different layer numbers in different areas of the board also enables us to model ‘flex-rigid’ boards, where rigid areas of board are connected by flexible layers.

When a board is defined in terms of layers, it is usual to position components relative to those layers using the ‘AssembleTo’ property, rather than at a specific point in 3D space. By positioning components relative to layers their position in the z direction will remain correct even if the thickness of the layers is changed.

Multiple Layer Stackups

Simple boards can be represented by a single layer stackup. First each layer is defined in IDX including properties such as its thickness and type. A layer stackup is then defined to describe the order those layers are “stacked” on top of each other to produce a cross-section such as shown in the image above.

In the case of more complex boards multiple layer stackups are common. The diagram below shows a cross-section of a typical ‘Flex-rigid’ board with different layer stackups for different areas of the board. The rigid area is represented by the ‘Primary Stackup’, then flexible areas by ‘Stackup 2’ and ‘Stackup 3’.

In this case the method of defining layers, stackups and zone is the same. First, each layer would be defined, then the three stackups would be defined in terms of those layers, and finally layer ‘zones’ are defined (which XY board areas are allocated a specific Stackup).

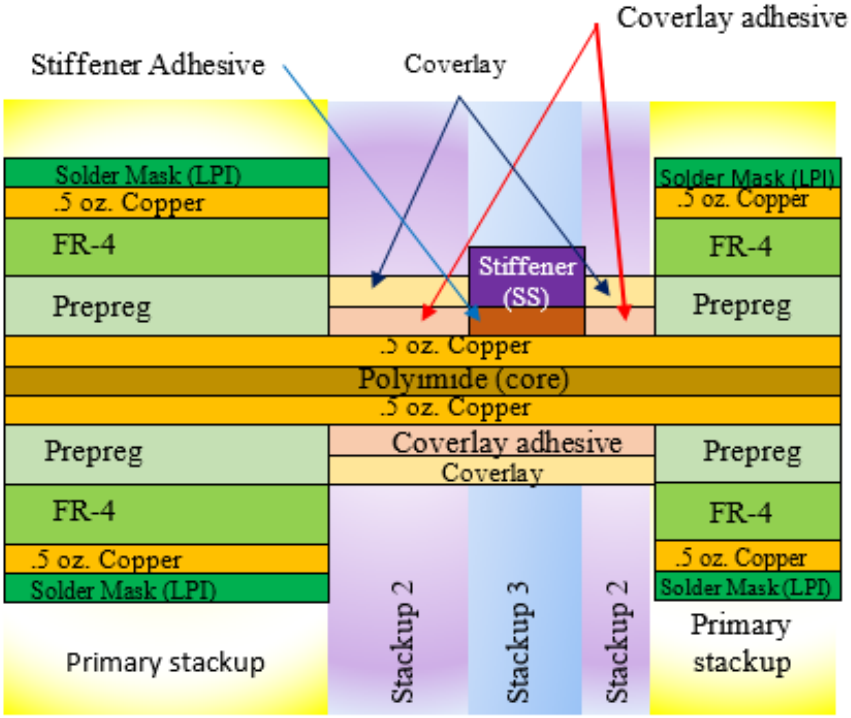


Figure 25: A cross-section of a typical flex-rigid board

Multiple Layer Zones

Note it is possible for two or more 2D XY Layer ‘Zones’ to occupy the same 2D space in some boards. For example, the flexible board shown in its ‘flexed’ state below would be described ‘flat’ in the IDX format.

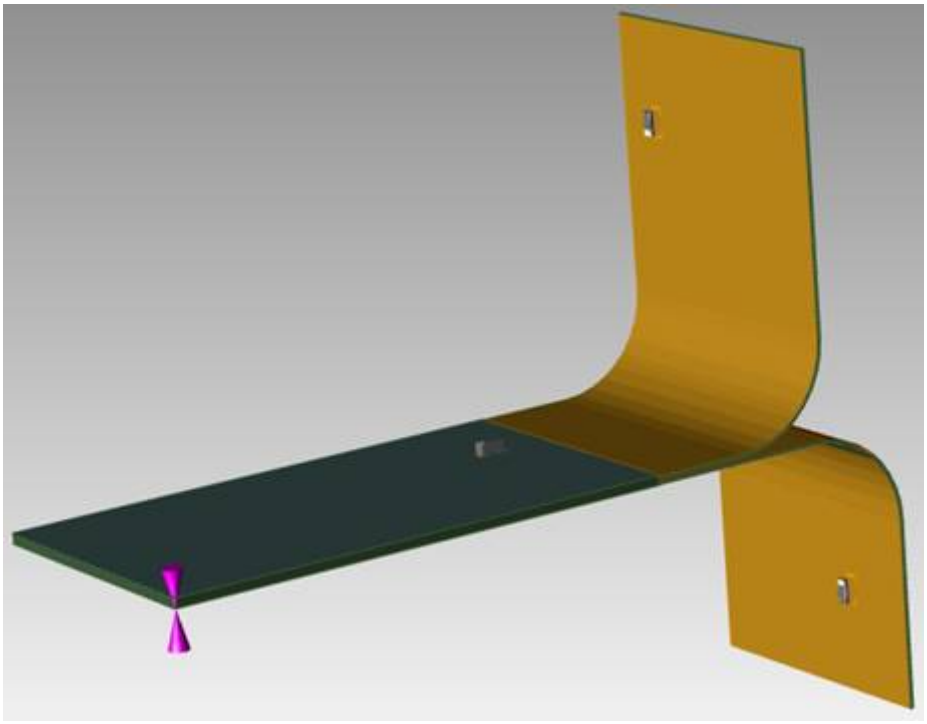


Figure 26: Example of a flexible board in its 'flexed' state

Both gold colored flexible areas shown (the one bent up and the one bent down would be described in their flat state in IDX, so both would have the same XY Zone definition, but be associated with two different layer stackups.

Physical vs Design Layers

The concept of layers is used within ECAD systems to group together items of a certain type which can then by hidden or shown system by simply hiding and showing that layer. Layers of this type are referred to in the IDX format as a Design Layer. Design Layers do not have a physical thickness and aren't part of the manufacturing process of the board, they are simply an aid to design. The Layer Stackup consists of Physical Layers, where a Physical Layer is a layer of conductive or insulating material with a thickness that can be stacked up to form the completed board.

Physical Layer Types

The following Physical Layer Types are currently supported in IDX. For IDXv4.0 and earlier a *UserProperty* called "LayerPurpose" is added either to the Item describing the Layer, or to the Instance of the Item describing the Layer within the Layer Stackup Item as detailed in the examples below.

In IDXv4.0 the Layer's geometryType attribute can be set directly as shown in the table below negating the requirement to add a *UserProperty*.

The UserProperty name and IDXv4.0 *geometryType* to use for each type of Physical Layer is given in the table below. Note that these values are the same as the *EDMDLayerPurpose* enumerator, except for 'Dielectric' which is not available in the IDXv3.0 schema.

UserProperty [LayerType]	IDXv4.0 GeometryType
SolderMask	LAYER_SOLDERMASK
SolderPaste	LAYER_SOLDERPASTE
SilkScreen	LAYER_SILKSCREEN
GenericLayer	LAYER_GENERIC
Glue	LAYER_GLUE
GlueMask	LAYER_GLUEMASK
PasteMask	LAYER_PASTEMASK
OtherSignal	LAYER_OTHERSIGNAL
LandsOnly	LAYER_LANDSONLY
PowerOrGround	LAYER_POWERORGROUND
EmbeddedPassiveCapacitorDielectric	LAYER_EMBEDDED_CAP_DIELECTRIC
EmbeddedPassiveResistor	LAYER_EMBEDDED_RESISTOR
Dielectric	LAYER_DIELECTRIC

Table 4: User property names and IDXv4.0 Geometry Types for Physical Layers

6.1.2.1 Defining Layers

Every board defined in terms of Layers requires at least one Layer and one or more Layer Stackups defined in the IDX file. The Layers can be defined as shown below:

1) First each physical Layer is defined as an *EDMDItem* of type assembly.

```
<!—First Layer. The geometryType attribute was introduced in IDX 4.0 and
specifies the Item is a Layer and gives its layer type-(LAYER_SIGNAL,
LAYER_POWER, etc.) -->


<foundation:Item id="ITEM0" geometryType=”LAYER_OTHERSIGNAL”
xsi:type="pdm:EDMDItem">
      <foundation:Name>INNER_Layer_1</foundation:Name>
      <foundation:Description>ActivePassivePCB</foundation:Description>
      <pdm:ItemType>assembly</pdm:ItemType>
      <!—Insert pdm:Identifier object containing this Layer's Identifier here --
>


       <!—Optional User Property specifiying the Layer’s type. This is not
required if the Layer type has already been specified by the geometryType
attribute above. If geometryType or this property isn’t present, then this
property must be specified in the Item representing the Stackup (below) -->


      <foundation:UserProperty IsChanged="false" IsNew="false"
xsi:type="property:EDMDUserSimpleProperty">
            <property:Key xsi:type="foundation:EDMDName">
                  <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
                  <foundation:ObjectName>LayerType</foundation:ObjectName>
             </property:Key>
            <property:Value>Signal</property:Value>
      </foundation:UserProperty>


      <!—Optional User Properties with each Layer’s Upper and Lower Bound -->
      <!—If the Upper and Lower Bounds aren’t specified here, they need to be
specified in the Stackup Item below -->


      <foundation:UserProperty xsi:type="property:EDMDUserSimpleProperty">
            <property:Key Persistant="false">
                  <foundation:SystemScope>SYSTEM_1</foundation:SystemScope>
                  <foundation:ObjectName>LowerBound</foundation:ObjectName>
            </property:Key>
            <property:Value>35.100000</property:Value>
      </foundation:UserProperty>
      <foundation:UserProperty xsi:type="property:EDMDUserSimpleProperty">
```

```
            <property:Key Persistant="false">
                  <foundation:SystemScope>SYSTEM_1</foundation:SystemScope>
                  <foundation:ObjectName>UpperBound</foundation:ObjectName>
            </property:Key>
            <property:Value>36.300000</property:Value>
      </foundation:UserProperty>

      <!—Finally add the name of this Layer as referenced by the Layer
Stackup(s) -->

      <pdm:ReferenceName>INNER_Layer_1</pdm:ReferenceName>
</foundation:Item>

<!— Second Layer --->

<foundation:Item id="ITEM1" xsi:type="pdm:EDMDItem">
      <foundation:Name>INNER_Layer_2</foundation:Name>
      <foundation:Description>ActivePassivePCB</foundation:Description>
      <pdm:ItemType>assembly</pdm:ItemType>
      <!--pdm:Identifier object containing this Layer's Identifier here -->
      <!—Specify LayerType, Upper and Lower Bounds here (as ITEM0 above) or in
the Stackup Item below -->
      <pdm:ReferenceName>INNER_Layer_2</pdm:ReferenceName>
</foundation:Item>

<!— Third Layer --->

<foundation:Item id="ITEM2" xsi:type="pdm:EDMDItem">
      <foundation:Name>INNER_Layer_3</foundation:Name>
      <foundation:Description>ActivePassivePCB</foundation:Description>
      <pdm:ItemType>assembly</pdm:ItemType>
      <!--pdm:Identifier object containing this Layer's Identifier -->
      <!—Specify Layer Type, Upper and Lower Bounds here (as ITEM0 above) or in
the Stackup Item below -->
      <pdm:ReferenceName>INNER_Layer_3</pdm:ReferenceName>
</foundation:Item>

<!-- ...and so on for each Physical Layer in the Board -->
```

6.1.2.2 Defining the Layer Stackup

Now the Layers are defined, we need to define one or more Layer Stackups as required. Each Layer Stackup references the Layers previously and defines their order in the stack:

```
<foundation:Item id="ITEM20" geometryType=”LAYER_STACKUP”
xsi:type="pdm:EDMDItem">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>ActivePassivePCB</foundation:Description>
      <pdm:ItemType>assembly</pdm:ItemType>
      <!—Include pdm:Identifier object containing this Layer's Identifier here -
->
        <!-- pdm:ItemInstance for each Layer in the Layer Stackup -->
      <!—First Layer in Stack: -->

      <pdm:ItemInstance xsi:type="pdm:EDMDItemInstance">

            <!—Optional UserProperty to specify the Layer type: Soldermask,
Signal, Dielectric etc. if not defined by the geometryType attribute or defined
previously in the Layer Stackup above -->

            <foundation:UserProperty IsChanged="false" IsNew="false"
xsi:type="property:EDMDUserSimpleProperty">
                  <property:Key xsi:type="foundation:EDMDName">

<foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
                        <foundation:ObjectName>LayerType</foundation:ObjectName>
                  </property:Key>
                  <property:Value>Soldermask</property:Value>
            </foundation:UserProperty>

            <!—UserProperties to specify the Layer’s Upper & Bound. Not required
if Upper & Lower Bounds of each Layer were specified in the Layer Stackup Item
above -->

             <foundation:UserProperty IsChanged="false" IsNew="false"
xsi:type="property:EDMDUserSimpleProperty">
                  <property:Key xsi:type="foundation:EDMDName">

<foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>

<foundation:ObjectName>UpperBound</foundation:ObjectName>
                  </property:Key>
                  <property:Value>0.069</property:Value>
```

```
            </foundation:UserProperty>
            <foundation:UserProperty IsChanged="false" IsNew="false"
xsi:type="property:EDMDUserSimpleProperty">
                  <property:Key xsi:type="foundation:EDMDName">

<foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>

<foundation:ObjectName>LowerBound</foundation:ObjectName>
                  </property:Key>
                  <property:Value>0.067</property:Value>
            </foundation:UserProperty>

            <pdm:InstanceName xsi:type="foundation:EDMDName">
                  <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
                  <foundation:ObjectName>INNER_Layer_1</foundation:ObjectName>
            </pdm:InstanceName>
            <pdm:Item>ITEM0</pdm:Item>    <!-- Reference to the Layer -->
      </pdm:ItemInstance>

      <!—Second Layer in the Stack: -->

      <pdm:ItemInstance xsi:type="pdm:EDMDItemInstance">

            <!—UserProperty specifying the Layer type. Required if not defined
in the Stackup previously -->

           <foundation:UserProperty IsChanged="false" IsNew="false"
xsi:type="property:EDMDUserSimpleProperty">
                <property:Key xsi:type="foundation:EDMDName">

<foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
                     <foundation:ObjectName>LayerType</foundation:ObjectName>
                </property:Key>
                <property:Value>Signal</property:Value>
            </foundation:UserProperty>

            <!—Upper and Lower Bounds Properties only required if not defined in
the Stackup previously -->

            <foundation:UserProperty IsChanged="false" IsNew="false"
xsi:type="property:EDMDUserSimpleProperty">
            <property:Key xsi:type="foundation:EDMDName">
                  <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
```

```
                    <foundation:ObjectName>UpperBound</foundation:ObjectName>
              </property:Key>
             <property:Value>0.067</property:Value>
             </foundation:UserProperty>
             <foundation:UserProperty IsChanged="false" IsNew="false"
xsi:type="property:EDMDUserSimpleProperty">
                    <property:Key xsi:type="foundation:EDMDName">

<foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
                          <foundation:ObjectName>LowerBound</foundation:ObjectName>
                    </property:Key>
                    <property:Value>0.0663</property:Value>
             </foundation:UserProperty>
              <pdm:InstanceName xsi:type="foundation:EDMDName">
                    <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
                    <foundation:ObjectName>INNER_Layer_2</foundation:ObjectName>
              </pdm:InstanceName>
              <pdm:Item>ITEM1</pdm:Item>          <!-- Reference to the Layer -->
       </pdm:ItemInstance>

       <!—Continue adding ItemInstances for each Layer in the Layer Stackup as
above -->

       <!—ReferenceName contains the name of this Layer Stackup as referenced by
the Layer Zones and/or Components etc. -->
       <pdm:ReferenceName>PCB</pdm:ReferenceName>
</foundation:Item>
```

6.1.2.3 Defining Layer Zones

A ‘Layer Zone’ can be thought of as an X-Y area of the board which has a certain Layer Stackup (Z). A simple board may have one Layer Stackup describing the ‘sandwich’ of layers in the board and one Layer Zone describing the whole surface area of the board which is associated with that Stackup.

More complex boards will have several Layer Stackups defined, and hence different ‘Zones’ of the board that are associated with different Layer Stackups. A flex-rigid board is a good example of this.

The figure below shows a typical flex-rigid board. The flexible areas of the board have less layers and hence less thickness than the rigid areas. The rigid and flexible areas of this board would be described in IDX using Layer Zones, with four zones representing the four rigid areas, and three zones representing the three flexible areas. There could be only two Layer Stackups – one for all the rigid areas and one for all the flexible areas, or there could be up to seven; one for each zone.

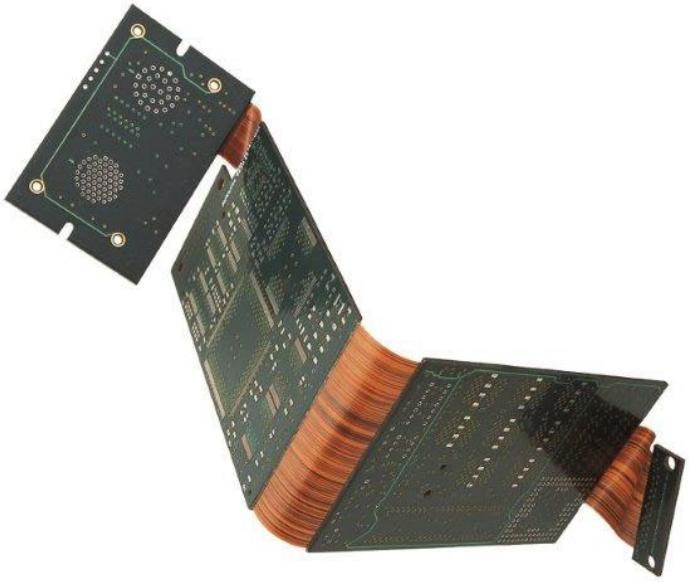


Figure 27 Connecting boards using flexible bands

The *FunctionalItemShapeType* property of the Layer Zone’s referenced *FunctionalItemShape* determines the function of each Layer Zone. Possible values for *FunctionalItemShapeType* are:

- *FlexibleArea* - a flexible area of the board.
- *RigidArea* - a rigid area of the board where components are placed.
- *Stiffener* - a closed shape that represents a section of the board that contains stiffening material. Certain constraints are implied as a result.

In the example image above, the flexible areas would be defined as having *FunctionalItemShapeType* of *FlexibleArea*. The rigid areas would normally be defined as having *FunctionalItemShapeType* of *RigidArea*, although any type other than *FlexibleArea* would be assumed to be rigid.

Layer Zones are very similar to Board Outline Items, and are described in a similar way, with the use of the *EDMDItem*’s ‘AssembleToName’ property indicating the Layer Stackup the Layer Zone is associated with as shown below:

1) A top level *EDMDItem* of type “assembly” represents the Layer Zone instance. The geometryType attribute specifies the Layer Zone/Board Area and gives its purpose.

```
 <foundation:Item id="ITEM57" geometryType=”BOARD_AREA_FLEXIBLE”
xsi:type="pdm:EDMDItem">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>NEW FLEXIBLE AREA</foundation:Description>
      <pdm:ItemType>assembly</pdm:ItemType>
      <!—Include pdm:Identifier object containing this Object’s Identifier here
-->
      <pdm:ItemInstance xsi:type="pdm:EDMDItemInstance">
            <pdm:InstanceName xsi:type="foundation:EDMDName">
                  <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>

<foundation:ObjectName>MGC::ExpeditionPCB_CORPORATE::207</foundation:ObjectName>
           </pdm:InstanceName>
            <pdm:Item>ITEM56</pdm:Item>
      </pdm:ItemInstance>

      <!—The ‘AssembleToName’ property refers to the LayerStackup related to
this area/zone-->

      <pdm:AssembleToName>layers_13_span_16</pdm:AssembleToName>
    </foundation:Item>
```

2) An *EDMDItem* of type “single” represents the Zone:

```
<foundation:Item id="ITEM56" xsi:type="pdm:EDMDItem">
      <foundation:Name>NEW FLEXIBLE AREA</foundation:Name>
      <foundation:Description>FLEXIBLE AREA GEOMETRY</foundation:Description>
      <pdm:ItemType>single</pdm:ItemType>
       <!—Include pdm:Identifier object containing this Object’s Identifier here
-->
      <pdm:Shape>FUNCTIONALITEMSHAPE2</pdm:Shape>
</foundation:Item>
```

3) The *EDMDItem*’s Shape references a *FunctionalItemShape* object. The *FunctionalItemShapeType* is used to indicate the purpose of the Zone as described above. This Object can be omitted if the geometryType attribute is present on the top level *EDMDItem* (e.g. so ITEM56’s Shape directly references SHAPE15 instead of FUNCTIONALITEMSHAPE2)

```
<foundation:FunctionalItemShape
id="FUNCTIONALITEMSHAPE2"xsi:type="pdm:EDMDFunctionalItemShape">
     <pdm:ShapeElement>SHAPE15</pdm:ShapeElement>
      <pdm:FunctionalItemShapeType>FlexibleArea</pdm:FunctionalItemShapeType>
```

```
      <!-- Other possible FunctionalItemShapeType values are UserArea,
MechanicalItem, Stiffener, or PlacementGroupArea -->

 </foundation:FunctionalItemShape>
```

4) Finally, the Shape is defined. Note that the upper and lower bound values in the referenced *CurveSet2d* should correspond to the upper and lower bounds of the associated Layer Stackup specified by the "AssembleToName" property of the parent *EDMDItem*:

```
<foundation:ShapeElement id="SHAPE15" xsi:type="pdm:EDMDShapeElement">
      <pdm:ShapeElementType>FeatureShapeElement</pdm:ShapeElementType>
      <pdm:Inverted>false</pdm:Inverted>
      <pdm:DefiningShape>CURVESET15</pdm:DefiningShape>
</foundation:ShapeElement>

<foundation:CurveSet2d id="CURVESET15" xsi:type="d2:EDMDCurveSet2d">
      <pdm:ShapeDescriptionType>GeometricModel</pdm:ShapeDescriptionType>
      <d2:LowerBound xsi:type="property:EDMDLengthProperty">
            <property:Value>0.004700</property:Value>
      </d2:LowerBound>
      <d2:UpperBound xsi:type="property:EDMDLengthProperty">
            <property:Value>0.021500</property:Value>
      </d2:UpperBound>

<d2:DetailedGeometricModelElement>POLYLN15</d2:DetailedGeometricModelElement>
</foundation:CurveSet2d>
```

6.1.2.4 Defining Bends in Flexible Boards

Bends in Flexible Areas are represented by two items in the IDX format. The first item represents the flexible area of the board. This is defined as shown in the previous section by defining a Layer Zone referencing a ShapeElement with the attribute *EDMDFunctionalItemShapeType* of *FlexibleArea*. A flexible area needs to be a closed shape.

The second item describes the area of the bend itself using a subtype of *EDMDBend* for the shape element of the item. In current version of schema there is only one subtype (*EDMDCircularBendType*) defined.

There is no check in the schema if a flexible area is described by an item with a "suitable" subtype of *EDMDItemShape*. This needs to be ensured by the ECAD/MCAD application.

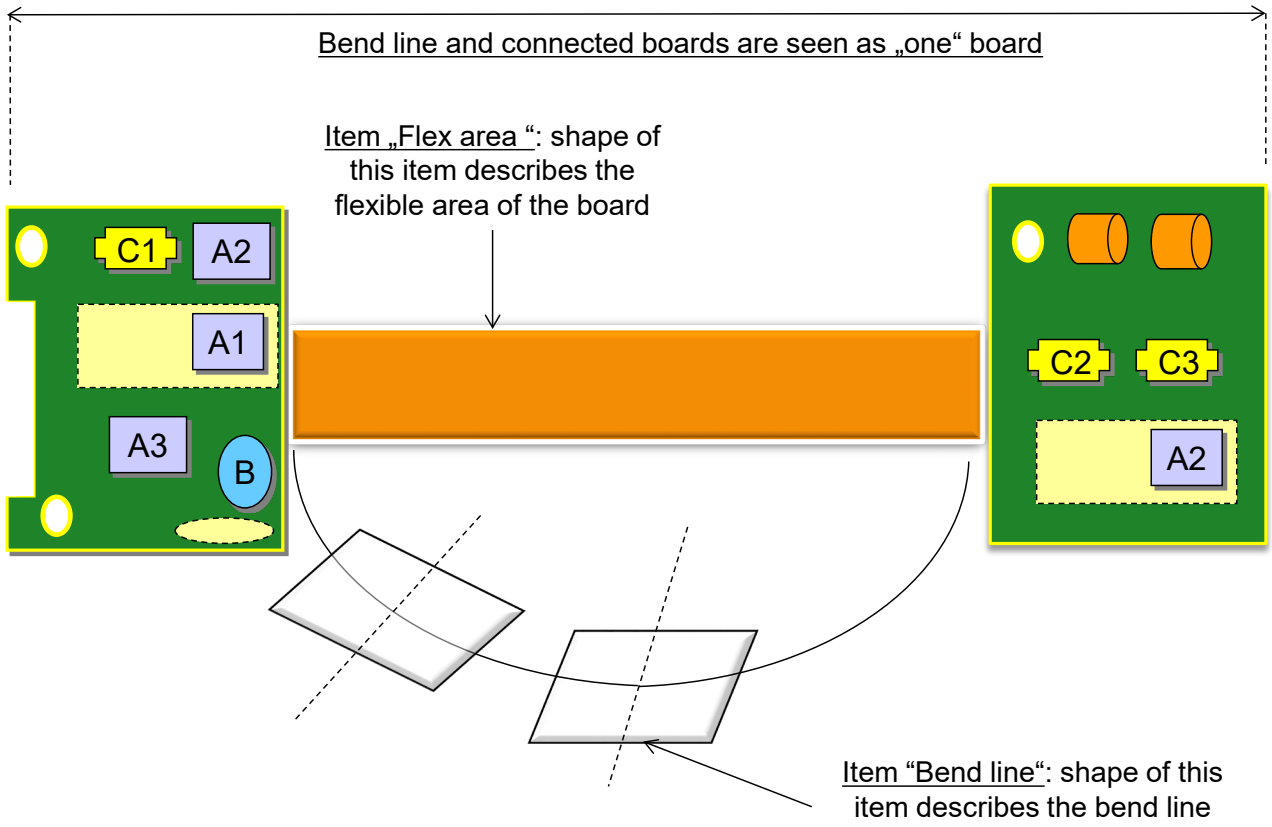


Figure 28: Describing a flexible area of board as bend

Seen here is an example of how to describe a Bend in terms of a Bend Line (the axis of the bend) and a Bend Ares (the 3D area that will be distorted by the Bend):

1) The top level *EDMDItem* of type "assembly" represents the Bend instance. Note the geometryType attribute confirming this item represents a bend, and the *bendSequenceNumber* attribute on the ItemInstance.

The *bendSequenceNumber* tells the ECAD/MCAD system the order to apply the bends when folding the board. Each consecutive *bendSequenceNumber* is assigned to one or more bends and specifies the sequence in which those bends should be applied when folding the board.

For example a board could have five bends; one with a bendSequenceNumber of 1 (which would be applied first), three with a bendSequenceNumber of 2 (which could be applied in any order after the first bend), and one with a bendSequenceNumber of 3 which would be the last to be applied.

If no *bendSequenceNumber* is specified it is assumed the bends can be applied in any order.

```
<foundation:Item id="ITEM1" geometryType=”BEND”  IsAttributeChanged="false">
     <foundation:Name> Bend12</foundation:Name>
     <foundation:Description> Bend12</foundation:Description>
     <pdm:ItemType>assembly</pdm:ItemType>
       <!-- pdm:Identifier object containing item's ID -->
      <pdm:ItemInstance bendSequenceNumber ="1">
           <pdm:Item>BENDITEM1</pdm:Item>
          <pdm:InstanceName IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
               <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
               <foundation:ObjectName>Bend12</foundation:ObjectName>
          </pdm:InstanceName>
     </pdm:ItemInstance>
</foundation:Item>
```

2) EDMDItem of type single represents the Bend:

```
<foundation:Item id="BENDITEM1" IsAttributeChanged="false">
      <!-- pdm:Identifier object containing item's ID here -->
      <foundation:Name> Bend12</foundation:Name>
      <foundation:Description> Bend12</foundation:Description>
      <pdm:ItemType>single</pdm:ItemType>
      <pdm:Shape>BEND1</pdm:Shape>
</foundation:Item>
```

3) Define the Bend itself. This provides information about the radius etc. and references a Bend Area and Bend Line that describe the geometry of the Bend:

```
<!-- Bend references both Bend Zone and Bend Line --->
<!-- Defines bend side, inner radius and angle --->
<foundation:Bend id=" BEND1" xsi:type="pdm:EDMDBend">
<foundation:Name>Bend Area</foundation:Name>
      <pdm:ShapeElement>SHAPE24</pdm:ShapeElement>
      <pdm:BendType xsi:type="pdm:EDMDCircularBendType">
            <pdm:BendLine>BENDLINE1</pdm:BendLine>
            <pdm:InnerSide>Top</pdm:InnerSide>
            <pdm:InnerRadius>6.350</pdm:InnerRadius>
            <pdm:InnerAngle>90.0</pdm:InnerAngle>
      </pdm:BendType>
</foundation:Bend>
```

3) The Bend Area (SHAPE24) is defined:

```
<!—Define the Bend Area: 4x 2D Points --->
<foundation:CartesianPoint id="PT133" xsi:type="d2:EDMDCartesianPoint">
      <d2:X xsi:type="property:EDMDLengthProperty">
            <property:Value>26.670000</property:Value>
      </d2:X>
      <d2:Y xsi:type="property:EDMDLengthProperty">
            <property:Value>61.044861</property:Value>
      </d2:Y>
</foundation:CartesianPoint>

<foundation:CartesianPoint id="PT134" xsi:type="d2:EDMDCartesianPoint">
      <d2:X xsi:type="property:EDMDLengthProperty">
            <property:Value>3.810000</property:Value>
      </d2:X>
      <d2:Y xsi:type="property:EDMDLengthProperty">
            <property:Value>61.016576</property:Value>
      </d2:Y>
</foundation:CartesianPoint>

<foundation:CartesianPoint id="PT135" xsi:type="d2:EDMDCartesianPoint">
      <d2:X xsi:type="property:EDMDLengthProperty">
            <property:Value>3.810000</property:Value>
      </d2:X>
      <d2:Y xsi:type="property:EDMDLengthProperty">
            <property:Value>73.531141</property:Value>
      </d2:Y>
</foundation:CartesianPoint>

<foundation:CartesianPoint id="PT136" xsi:type="d2:EDMDCartesianPoint">
      <d2:X xsi:type="property:EDMDLengthProperty">
            <property:Value>26.670000</property:Value>
      </d2:X>
      <d2:Y xsi:type="property:EDMDLengthProperty">
            <property:Value>73.559426</property:Value>
      </d2:Y>
</foundation:CartesianPoint>


<!—Now define a closed 2D rectangular zone in terms of 2D points above -->
```

```
<foundation:PolyLine id="POLYLN31" xsi:type="d2:EDMDPolyLine">
      <d2:Point>PT133</d2:Point>
      <d2:Point>PT134</d2:Point>
      <d2:Point>PT135</d2:Point>
      <d2:Point>PT136</d2:Point>
      <d2:Point>PT133</d2:Point>
</foundation:PolyLine>

<!—Define a Curveset2d to add a z-Range to the rectangular zone defined above --
>
 <foundation:CurveSet2d id="CURVESET24" xsi:type="d2:EDMDCurveSet2d">
      <pdm:ShapeDescriptionType>GeometricModel</pdm:ShapeDescriptionType>
      <d2:LowerBound xsi:type="property:EDMDLengthProperty">
            <property:Value>1.678940</property:Value>
      </d2:LowerBound>
      <d2:UpperBound xsi:type="property:EDMDLengthProperty">
            <property:Value>1.968500</property:Value>
      </d2:UpperBound>

<d2:DetailedGeometricModelElement>POLYLN31</d2:DetailedGeometricModelElement>
</foundation:CurveSet2d>

<!—Define this the Bend Area as ShapeElement of type NonFeatureShapeElement --->
<!—We'll reference this area and the bend line in our bend later --->

<foundation:ShapeElement id="SHAPE24" xsi:type="pdm:EDMDShapeElement">
      <pdm:ShapeElementType>NonFeatureShapeElement</pdm:ShapeElementType>
      <pdm:Inverted>false</pdm:Inverted>
      <pdm:DefiningShape>CURVESET24</pdm:DefiningShape>
</foundation:ShapeElement>
```

4) Now define the Bend Line (BENDLINE1):

```
<!-- Bend Line consists of 2x 2D Points --->

<foundation:CartesianPoint id="PT138" xsi:type="d2:EDMDCartesianPoint">
      <d2:X xsi:type="property:EDMDLengthProperty">
            <property:Value>-4.445000</property:Value>
      </d2:X>
      <d2:Y xsi:type="property:EDMDLengthProperty">
```

```
            <property:Value>67.263645</property:Value>
      </d2:Y>
</foundation:CartesianPoint>

<foundation:CartesianPoint id="PT139" xsi:type="d2:EDMDCartesianPoint">
      <d2:X xsi:type="property:EDMDLengthProperty">
            <property:Value>33.020000</property:Value>
      </d2:X>
      <d2:Y xsi:type="property:EDMDLengthProperty">
            <property:Value>67.310000</property:Value>
      </d2:Y>
</foundation:CartesianPoint>

<!—Now define the Bend Line in terms of 2D Points above -->

<foundation:Line id="BENDLINE1" xsi:type="d2:EDMDLine">
      <d2:Point>PT138</d2:Point>
      <d2:Vector>PT139</d2:Vector>
</foundation:Line>
```

6.1.2.5 Positioning Items in terms of Layers

Components, Keepouts and other items can be positioned in two ways in IDX; either at an absolute location in 3D space as defined by a 3D transformation matrix, or by specifiying the Layer they are mounted to and with a 2D transformation matrix to describe x,y and rotation and an optional z-offset from the layer surface.

The latter method is more flexible, as any change to the board layer thickness or stackup will not subsequently require the component positions to be recalculated.

An *EDMDItem*'s "AssembleToName" property specifies a reference Layer or Surface for an item positioned in terms of a Layer.

The following example shows a Component Instance positioned relative to the Layer 'INNER_Layer_9':

```
<foundation:Item id="ITEM41" geometryType="COMPONENT" xsi:type="pdm:EDMDItem">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>NEW BOARD DESCRIPTION</foundation:Description>
      <pdm:ItemType>assembly</pdm:ItemType>
       <!—Include pdm:Identifier object containing this Object's Identifier here
-->
      <pdm:ItemInstance xsi:type="pdm:EDMDItemInstance">
            <foundation:UserProperty IsChanged="false" IsNew="false"
xsi:type="property:EDMDUserSimpleProperty">
                  <property:Key xsi:type="foundation:EDMDName">

<foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
                        <foundation:ObjectName>REFDES</foundation:ObjectName>
                  </property:Key>
                <property:Value>C11</property:Value>
          </foundation:UserProperty>
          <foundation:UserProperty IsChanged="false" IsNew="false"
xsi:type="property:EDMDUserSimpleProperty">
                <property:Key xsi:type="foundation:EDMDName">

<foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
                      <foundation:ObjectName>PARTNUM</foundation:ObjectName>
                </property:Key>
                <property:Value>12333-CAP</property:Value>
          </foundation:UserProperty>
          <foundation:UserProperty IsChanged="false" IsNew="false"
xsi:type="property:EDMDUserSimpleProperty">
                <property:Key xsi:type="foundation:EDMDName">

<foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
                      <foundation:ObjectName>PKGNAME</foundation:ObjectName>
                </property:Key>
                <property:Value>CC1206</property:Value>
```

```
          </foundation:UserProperty>
          <pdm:InstanceName xsi:type="foundation:EDMDName">
                <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>

<foundation:ObjectName>MGC::ExpeditionPCB_CORPORATE::196</foundation:ObjectName>
          </pdm:InstanceName>

         <!—2D transformation describes x,y position and offset -->

          <pdm:Transformation xsi:type="pdm:EDMDTransformation">
                <pdm:TransformationType>d2</pdm:TransformationType>
                <pdm:xx>1.000000</pdm:xx>
                <pdm:xy>0.000000</pdm:xy>
                <pdm:yx>0.000000</pdm:yx>
                <pdm:yy>1.000000</pdm:yy>
                <pdm:zx>0.000000</pdm:zx>
                <pdm:zy>0.000000</pdm:zy>
                <pdm:tx xsi:type="property:EDMDLengthProperty">
                      <property:Value>3.160000</property:Value>
                </pdm:tx>
                <pdm:ty xsi:type="property:EDMDLengthProperty">
                      <property:Value>0.845000</property:Value>
                </pdm:ty>
          </pdm:Transformation>
          <pdm:Item>ITEM40</pdm:Item>
    </pdm:ItemInstance>

   <!—AssembleToName references a Layer. Use this to get Item’s z-location -->
   <!—Use the surface of the Layer furthest from the center of the board as the
reference -->

    <pdm:AssembleToName>INNER_Layer_9</pdm:AssembleToName>
</foundation:Item>
```

6.1.2.6 Defining Items that span layers

Some items such as holes and vias may span one or more physical layers. In this case the same *AssembleToName* attribute is used, but a layer stack-up is referenced rather than a single layer. For example, a hole may span two layers as described below:

```
<!-- Define the Layer Stack-up with the name 'Layers_1_to_2-->

<foundation:Item id="ITEM20" geometryType="LAYER_STACKUP"
xsi:type="pdm:EDMDItem">
      <pdm:ReferenceName>Layers_1_to_2</pdm:ReferenceName>
      <pdm:ItemType>assembly</pdm:ItemType>
      <!-- First Layer in Stack: -->
      <pdm:ItemInstance xsi:type="pdm:EDMDItemInstance">
            <pdm:InstanceName xsi:type="foundation:EDMDName">
                  <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
                  <foundation:ObjectName>Layer_1</foundation:ObjectName>
            </pdm:InstanceName>
            <pdm:Item>ITEM0</pdm:Item>
      </pdm:ItemInstance>
      <!-- –Second Layer in the Stack: -->
      <pdm:ItemInstance xsi:type="pdm:EDMDItemInstance">
            <pdm:InstanceName xsi:type="foundation:EDMDName">
                  <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
                 <foundation:ObjectName>Layer_2</foundation:ObjectName>
            </pdm:InstanceName>
            <pdm:Item>ITEM1</pdm:Item>
      </pdm:ItemInstance>
</foundation:Item>

<!-- Now define the geometry of the hole --->

<foundation:CurveSet2d id="CURVESETMH" xsi:type="d2:EDMDCurveSet2d">
     <pdm:ShapeDescriptionType>GeometricModel</pdm:ShapeDescriptionType>
     <d2:LowerBound xsi:type="property:EDMDLengthProperty">
          <property:Value>0.742</property:Value>
     </d2:LowerBound>
     <d2:UpperBound xsi:type="property:EDMDLengthProperty">
         <property:Value>1.144</property:Value>
     </d2:UpperBound>
</foundation:CurveSet2d>

<foundation:ShapeElement id="SHAPEMH" xsi:type="pdm:EDMDShapeElement">
     <pdm:ShapeElementType>FeatureShapeElement</pdm:ShapeElementType>
     <pdm:Inverted>false</pdm:Inverted>
     <pdm:DefiningShape>CURVESETMH</pdm:DefiningShape>
```

```
</foundation:ShapeElement>

<foundation:Stratum id="STRATUMMH" xsi:type="pdm:EDMDStratum">
     <pdm:ShapeElement>SHAPEMH</pdm:ShapeElement>
     <pdm:StratumType>DesignLayerStratum</pdm:StratumType>
</foundation:Stratum>

<foundation:InterStratumFeature id="MHOLE0"
xsi:type="pdm:EDMDInterStratumFeature">
     <pdm:ShapeElement>SHAPE1</pdm:ShapeElement>
     <pdm:InterStratumFeatureType>PlatedCutout</pdm:InterStratumFeatureType>
     <pdm:Stratum>STRATUMMH</pdm:Stratum>
</foundation:InterStratumFeature>

<foundation:Item id="ITEM2" xsi:type="pdm:EDMDItem">
     <foundation:Name>Hole</foundation:Name>
     <foundation:Description>Reusable Padstack</foundation:Description>
     <pdm:ItemType>single</pdm:ItemType>
     <pdm:Identifier xsi:type="foundation:EDMDIdentifier">
          <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
          <foundation:Number>MH-6-H3.5mm-0</foundation:Number>
          <foundation:Version>1</foundation:Version>
          <foundation:Revision>0</foundation:Revision>
          <foundation:Sequence>1</foundation:Sequence>
     </pdm:Identifier>
     <pdm:Shape>MHOLE0</pdm:Shape>
</foundation:Item>

<!-- Top level Item of type 'Assembly' references the Layer Stack-up -->

<foundation:Item id="ITEM3" xsi:type="pdm:EDMDItem">
     <foundation:Name>PCB</foundation:Name>
     <foundation:Description>NEW BOARD DESCRIPTION</foundation:Description>
     <pdm:ItemType>assembly</pdm:ItemType>
     <!-- Item identifier here -->
     <pdm:ItemInstance xsi:type="pdm:EDMDItemInstance">
          <foundation:UserProperty IsChanged="false" IsNew="false"
type="property:EDMDUserSimpleProperty">
               <property:Key xsi:type="foundation:EDMDName">
                    <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
                    <foundation:ObjectName>MHNAME</foundation:ObjectName>
```

```
                </property:Key>
                <property:Value>MHole</property:Value>
            </foundation:UserProperty>
            <pdm:InstanceName xsi:type="foundation:EDMDName">
                 <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
                 <foundation:ObjectName>MGC::ExpeditionPCB
V10::16</foundation:ObjectName>
            </pdm:InstanceName>
            <pdm:Transformation xsi:type="pdm:EDMDTransformation">
                 <pdm:TransformationType>d3</pdm:TransformationType>
                 <pdm:xx>1.000000</pdm:xx>
                 <pdm:xy>0.000000</pdm:xy>
                 <pdm:xz>0.000000</pdm:xz>
                 <pdm:yx>0.000000</pdm:yx>
                 <pdm:yy>1.000000</pdm:yy>
                 <pdm:yz>0.000000</pdm:yz>
                 <pdm:zx>0.000000</pdm:zx>
                 <pdm:zy>0.000000</pdm:zy>
                 <pdm:zz>1.000000</pdm:zz>
                 <pdm:tx xsi:type="property:EDMDLengthProperty">
                     <property:Value>37.051387</property:Value>
                 </pdm:tx>
                 <pdm:ty xsi:type="property:EDMDLengthProperty">
                     <property:Value>69.521618</property:Value>
                 </pdm:ty>
                 <pdm:tz xsi:type="property:EDMDLengthProperty">
                     <property:Value>0.000000</property:Value>
                 </pdm:tz>
            </pdm:Transformation>
            <pdm:Item>ITEM2</pdm:Item>
       </pdm:ItemInstance>
       <pdm:BaseLine>
            <property:Value>true</property:Value>
       </pdm:BaseLine>
       <!-- Reference a single Layer Stack-up Item rather than the individual
layers --->
       <pdm:AssembleToName>Layers_1_to_2</pdm:AssembleToName>
</foundation:Item>
```

6.2 How to model Components

Components can be defined in two ways in IDX; the 'traditional' way used by all versions of IDX and in a simplified form introduced in IDXv4.0.

Components are described by an *EDMDItem* which represents the instance(s) of the component, which references an *EDMDItem* describing the component itself, which then references an *EDMDAssemblyComponentobject*. The *EDMDAssemblyComponent* object defines the item as either a Mechanical or Electrical component, and in turn references the component geometry which is defined as an *EDMDCurveSet2D*.

An *EDMDAssemblyComponent.AssemblyComponentType* of "Physical" is used to represent electrical components, and an *EDMDAssemblyComponent.AssemblyComponentType* of "MechanicalItem" is used to represent mechanical components.

The simplified form of representing a mounting hole in IDXv4.0 allows the user to omit the *EDMDAssemblyComponent* providing they have specified a geometryType attribute of COMPONENT or COMPONENT_ MECHANICAL on the top level *EDMDItem*. Examples of both methods of describing a component are given below:

6.2.1.1 Describing a Component in IDX: "Traditional" method

1) Two *EDMDItems* of itemType assembly: Represent Component instances C1 & C2:

```
<!-- Instance C1: References Item CMPITEM209-->

<foundation:Item id="ITEM210" IsAttributeChanged="false">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>BOARD ASSEMBLY</foundation:Description>
      <pdm:ItemType>assembly</pdm:ItemType>
       <!-- pdm:Identifier object containing item's ID -->
      <pdm:ItemInstance>
           <pdm:Item>CMPITEM209</pdm:Item>
           <pdm:InstanceName IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
           <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
           <foundation:ObjectName>cc1210_pn-cc1210-1</foundation:ObjectName>
           </pdm:InstanceName>
           <foundation:Name>C1</foundation:Name>
          <!—Transformation can be 2D or 3D. -->
          <pdm:Transformation>
               <pdm:TransformationType>d2</pdm:TransformationType>
               <pdm:xx>0.62347</pdm:xx>
               <pdm:xy>-0.781847</pdm:xy>
               <pdm:yx>0.781847</pdm:yx>
               <pdm:yy>0.62347</pdm:yy>
               <pdm:tx IsAttributeChanged="false">
```

```
                    <property:Value>36.13551</property:Value>
               </pdm:tx>
               <pdm:ty IsAttributeChanged="false">
                    <property:Value>20.598426</property:Value>
               </pdm:ty>
          </pdm:Transformation>
     </pdm:ItemInstance>
</foundation:Item>


<!-- Instance C2 References Item CMPITEM209-->

<foundation:Item id="ITEM211" IsAttributeChanged="false">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>BOARD ASSEMBLY</foundation:Description>
      <pdm:ItemType>assembly</pdm:ItemType>
       <!-- pdm:Identifier object containing item's ID -->
      <pdm:ItemInstance>
           <pdm:Item>CMPITEM209</pdm:Item>
           <pdm:InstanceName IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
           <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
           <foundation:ObjectName>cc1210_pn-cc1210-2</foundation:ObjectName>
           </pdm:InstanceName>
           <foundation:Name>C2</foundation:Name>
           <pdm:Transformation>
           <pdm:TransformationType>d2</pdm:TransformationType>
               <pdm:xx>-0.22257</pdm:xx>
               <pdm:xy>-0.974917</pdm:xy>
               <pdm:yx>0.974917</pdm:yx>
               <pdm:yy>-0.22257</pdm:yy>
               <pdm:tx IsAttributeChanged="false">
                    <property:Value>78.436047</property:Value>
               </pdm:tx>
               <pdm:ty IsAttributeChanged="false">
                     <property:Value>10.943604</property:Value>
               </pdm:ty>
           </pdm:Transformation>
     </pdm:ItemInstance>
</foundation:Item>
```

2) *EDMDItem* of itemType single: Represents the Component itself:

```
<foundation:Item id="CMPITEM209" IsAttributeChanged="false">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>ELECTRIC COMPONENT
GEOMETRY</foundation:Description>
      <!-- pdm:Identifier object containing item's ID -->
      <!--User Property containing Component's Part Number -->
      <foundation:UserProperty xsi:type="property:EDMDUserSimpleProperty"
IsChanged="false" IsNew="false">
           <property:Key IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
                <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
                <foundation:ObjectName>PARTNUM</foundation:ObjectName>
           </property:Key>
           <property:Value>pn-cc1210</property:Value>
      </foundation:UserProperty>
      <pdm:ItemType>single</pdm:ItemType>
      <!—Component's Package Name -->
      <pdm:PackageName IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
            <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
            <foundation:ObjectName>cc1210</foundation:ObjectName>
      </pdm:PackageName>
      <pdm:Shape>COMPONENT3</pdm:Shape>
    </foundation:Item>
```

3) *AssemblyComponent* Object: *AssemblyComponentType* of "Physical" specifies that item is an Electrical Component.

```
<!-- Tells us the Shape is an Electrical Component -->

<foundation:AssemblyComponent id="COMPONENT3" IsAttributeChanged="false">
     <pdm:ShapeElement>CMPSHAPE102</pdm:ShapeElement>
     <pdm:AssemblyComponentType>Physical</pdm:AssemblyComponentType>  <!—Or
MechanicalItem -->
</foundation:AssemblyComponent>
```

4) *ShapeElement* CMPSHAPE102 describes the shape of the Component:

```
<!-- The Component's Shape -->
<foundation:ShapeElement id="CMPSHAPE102" IsAttributeChanged="false">
```

```
      <pdm:ShapeElementType>FeatureShapeElement</pdm:ShapeElementType>
      <pdm:DefiningShape>CMPCURVESET102</pdm:DefiningShape>
      <pdm:Inverted>false</pdm:Inverted>
</foundation:ShapeElement>
```

6.2.1.2 Describing a Component in IDX: “Simplified” method

The following XML example shows how to represent a Component using the ‘simplified’ method introduced in IDXv4.0. By specifying the initial item’s geometryType as COMPONENT or COMPONENT_MECHANICAL we can omit the *EDMDAssemblyComponent* object:

1) Two *EDMDItems* of *itemType* assembly: Represent Component instances C1 & C2 as before:

```
<!-- Instance C1: References Item CMPITEM209-->

<foundation:Item id="ITEM210" geometryType=”COMPONENT”>
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>BOARD ASSEMBLY</foundation:Description>
      <pdm:ItemType>assembly</pdm:ItemType>
       <!-- pdm:Identifier object containing item's ID -->
      <pdm:ItemInstance>
           <pdm:Item>CMPITEM209</pdm:Item>
           <pdm:InstanceName IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
           <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
           <foundation:ObjectName>cc1210_pn-cc1210-1</foundation:ObjectName>
           </pdm:InstanceName>
           <foundation:Name>C1</foundation:Name>
          <!—Transformation can be 2D or 3D. -->
          <pdm:Transformation>
               <pdm:TransformationType>d2</pdm:TransformationType>
               <pdm:xx>0.62347</pdm:xx>
               <pdm:xy>-0.781847</pdm:xy>
               <pdm:yx>0.781847</pdm:yx>
               <pdm:yy>0.62347</pdm:yy>
               <pdm:tx IsAttributeChanged="false">
                   <property:Value>36.13551</property:Value>
            </pdm:tx>
            <pdm:ty IsAttributeChanged="false">
                 <property:Value>20.598426</property:Value>
            </pdm:ty>
        </pdm:Transformation>
     </pdm:ItemInstance>
</foundation:Item>


<!-- Instance C2 References Item CMPITEM209-->
```

```
<foundation:Item id="ITEM211" geometryType=”COMPONENT”>
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>BOARD ASSEMBLY</foundation:Description>
      <pdm:ItemType>assembly</pdm:ItemType>
       <!-- pdm:Identifier object containing item's ID -->
      <pdm:ItemInstance>
           <pdm:Item>CMPITEM209</pdm:Item>
           <pdm:InstanceName IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
           <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
           <foundation:ObjectName>cc1210_pn-cc1210-2</foundation:ObjectName>
           </pdm:InstanceName>
           <foundation:Name>C2</foundation:Name>
           <pdm:Transformation>
           <pdm:TransformationType>d2</pdm:TransformationType>
               <pdm:xx>-0.22257</pdm:xx>
               <pdm:xy>-0.974917</pdm:xy>
               <pdm:yx>0.974917</pdm:yx>
               <pdm:yy>-0.22257</pdm:yy>
               <pdm:tx IsAttributeChanged="false">
                    <property:Value>78.436047</property:Value>
               </pdm:tx>
               <pdm:ty IsAttributeChanged="false">
                     <property:Value>10.943604</property:Value>
               </pdm:ty>
           </pdm:Transformation>
     </pdm:ItemInstance>
</foundation:Item>
```

2) *EDMDItem* of itemType single: Represents the Component itself:

```
<foundation:Item id="CMPITEM209">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>ELECTRIC COMPONENT
GEOMETRY</foundation:Description>
      <!-- pdm:Identifier object containing item's ID -->
      <!--User Property containing Component’s Part Number -->
      <foundation:UserProperty xsi:type="property:EDMDUserSimpleProperty"
IsChanged="false" IsNew="false">
```

```
            <property:Key IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
                  <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
                  <foundation:ObjectName>PARTNUM</foundation:ObjectName>
            </property:Key>
            <property:Value>pn-cc1210</property:Value>
      </foundation:UserProperty>
      <pdm:ItemType>single</pdm:ItemType>
      <!—Component’s Package Name -->
      <pdm:PackageName IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
             <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
             <foundation:ObjectName>cc1210</foundation:ObjectName>
      </pdm:PackageName>
      <pdm:Shape> CMPSHAPE102 </pdm:Shape>
    </foundation:Item>
```

3) The AssemblyComponent object is omitted and the ShapeElement object CMPSHAPE102 which describes the shape of the Component is referenced directly:

```
<!-- The Component's Shape -->
<foundation:ShapeElement id="CMPSHAPE102" IsAttributeChanged="false">
      <pdm:ShapeElementType>FeatureShapeElement</pdm:ShapeElementType>
      <pdm:DefiningShape>CMPCURVESET102</pdm:DefiningShape>
      <pdm:Inverted>false</pdm:Inverted>
</foundation:ShapeElement>
```

6.2.1.3 Associating a 3D CAD model with a component

Some Electrical and most Mechanical CAD systems are able to generate a 3D representation of the board and its components.

The IDX specification contains enough information to define a simple component shape consisting of 2D profiles extruded to a height. This "2½D" representation is suitable for applications where the only concern is the component's basic bounding box, but for some applications a fully detailed 3D model may be required for more detail.

The IDXv4.0 format supports detailed 3D model information with the *Model3D* object. One or more Model3D objects are added as a child element to the *foundation:Body* section of the IDX file and referenced by *EDMDItems* representing components. This approach of using references means that it's possible for several *EDMDItems* to reference the same *Model3D* object if required.

A *Model3D* object is defined as shown:

```
<foundation:Model3D id="RES123">
                <pdm:ModelIdentifier>Resistor.sldprt</pdm:ModelIdentifier>
<!--Model's filename or id for cloud CAD --->
                <pdm:ModelVersion>default</pdm:ModelVersion>
<!-- Optional Model's configuration/version id --->
                <pdm:ModelLocation>/Library/Resistors/</pdm: ModelLocation>
<!-- Optional location relative to root --->
                <pdm:MCADFormat>SolidWorks</pdm:MCADFormat>
<!-- Model's filetype - SolidWorks, NX --->
                <pdm:MCADFormatVersion>2019</pdm:MCADFormatVersion>
<!— Optional MCAD version. E.g. SolidWorks 2019 --->
                <pdm:Transformation></pdm:Transformation>
<!-- 2D or 3D transform translation needed to align --->
                <pdm:TransformationReference>CWX</pdm: TransformationReference>
<!—Optional reference for an MCAD coordinate system/feature needed to align the
model as an alternative to using a 2D/3D transform--->
</pdm:3DModel>
```

The properties of the 3DModel object are:

- **ModelIdentifier** – The 3D model's filename including suffix if it is a CAD file, e.g. *"Resistor.PRT", "Capacitor.sldprt"* or *"0402.step".* Alternatively, this could be a document ID if the model is in a web-based CAD system, cloud storage or PDM/PLM system.

- **ModelVersion** – An optional version or configuration ID for systems where one file can contain multiple configurations, versions, or representations.

- **ModelLocation** – An optional location relative to the root file storage location. It is assumed both ECAD and MCAD know the root storage location of the 3D models.

- **MCADFormat** – The file format of the 3D model. Any text value is acceptable, but it is recommended the following standard names are used for common MCAD formats:
  - *SolidWorks*
  - *SolidEdge*
  - *NX*
  - *Catia*
  - *STEP*
  - *OBJ*
  - *STL*
  - *Inventor*
  - *Fusion*

- *Onshape*

- **MCADFormatVersion** – Optional file format version. For example, *"2019"* for SolidWorks 2019 files or *"AP214"* for STEP AP214 files.

- **Transformation** – Contains an optional IDX 'd2' or 'd3' transformation object required to align the 3D model with the footprint shape described in the file. In many cases the origin of a 3D model may not align with the origin of the ECAD footprint. The ECAD footprint may assume a "pin one" origin for example, while the 3D model may have its origin in the center. Applying this transform to the 3D model will bring its position in line with that of the ECAD footprint. For simple (x,y) translations a d2 matrix can be used, for more complex transformations (such as those where the 3D model is aligned with a different 2D plane) a d3 matrix will be required.

- **TransformationReference** – Some systems may align 3D models by using reference geometry instead such as a user-defined coordinate system to represent the model's origin and alignment. In this case the Transformation field can be omitted and a text reference supplied in this field instead. "CWX" could mean to use the coordinate system "CWX" to align the model for example.

The 3DModel object is referenced by the optional *EDMD3DModel* attribute of the *EDMDItem* of itemType single representing the Component as shown:

```
<foundation:Item id="CMPITEM209">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>ELECTRIC COMPONENT
GEOMETRY</foundation:Description>
      <!-- pdm:Identifier object containing item's ID -->
      <!--User Property containing Component's Part Number -->
      <foundation:UserProperty xsi:type="property:EDMDUserSimpleProperty"
IsChanged="false" IsNew="false">
           <property:Key IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
                <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
                <foundation:ObjectName>PARTNUM</foundation:ObjectName>
           </property:Key>
           <property:Value>pn-cc1210</property:Value>
      </foundation:UserProperty>
      <pdm:ItemType>single</pdm:ItemType>
      <!—Component's Package Name -->
      <pdm:PackageName IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
            <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
            <foundation:ObjectName>cc1210</foundation:ObjectName>
      </pdm:PackageName>
      <pdm:Shape> CMPSHAPE102 </pdm:Shape>
     <pdm:EDMD3DModel>RES123</pdm: EDMD3DModel >
    </foundation:Item>
```

6.2.1.4 Z-Offset for Item instances

Introduction

Item instances in the IDX format (such as Component Instances) can be positioned using either a 2D or 3D transformation matrix. The 3D transform supports transformation of the instances in the z direction but is only suitable for defining an absolute z position, rather than a position relative to a layer or board surface.

Consequently, the 2D transformation matrix is usually used to define the position of a Component Instance relative to the upper or lower surface of a Board or Layer, but versions of IDX prior to version 4.0 don't allow any mechanism to offset an item instance in the z-direction relative to a Board or Layer surface.

IDXv4.0 introduces support for an optional 'zOffset' attribute on the *<pdm:ItemInstance>* object to address this issue. Any value specified z-offset-translation must be applied after the transformation to offset the item instance – the transform must first be applied to rotate and transform the instance to its initial location, then the z-offset is added (or subtracted if negative) from the instance's z-location.

Note that positive z offset is always defined as *away from* the surface of the specified layer or board surface in the direction of the z normal. Hence a z offset of +2mm would have the effect of moving components 2mm away from either the board's lower or upper surface. Similarly, an offset of -1mm would move components down 1mm into the board on either the upper or lower surface.

How to represent z-offset in IDXv4.0 file:

The following XML illustrates the usage of zOffset attribute (highlighted). In this example, component instance U14 is offset 2mm from the upper surface of the Layer *INNER_Layer_13*. Some XML elements have been removed for clarity:

```
<foundation:Item id="ITEM_4">
     <pdm:ItemType>assembly</pdm:ItemType>
     <pdm:Identifier Persistant="false">
     </pdm:Identifier>
     <pdm:ItemInstance id="ITEMINST_3" zOffset ="0.20">
         <foundation:Name>U14</foundation:Name>
         <pdm:InstanceName Persistant="false">
         </pdm:InstanceName>
         <pdm:Transformation>
             <pdm:TransformationType>2d</pdm:TransformationType>
             <pdm:xx>-0.707106781187</pdm:xx>
             <pdm:xy>0.707106781187</pdm:xy>
             <pdm:yx>0.707106781187</pdm:yx>
             <pdm:yy>0.707106781187</pdm:yy>
             <pdm:tx>
                 <property:Value>1100.00</property:Value>
             </pdm:tx>
             <pdm:ty>
                 <property:Value>1350.00</property:Value>
             </pdm:ty>
         </pdm:Transformation>
         <pdm:Item>SYMBOL_2</pdm:Item>
     </pdm:ItemInstance>
     <pdm:BaseLine>
   </pdm:BaseLine>
   <pdm:AssembleToName>INNER_Layer_13</pdm:AssembleToName>
 </foundation:Item>
```

6.2.1.5 How to define Component Pins from IDXv4.0

Introduction

Defining component pin shapes and positions has two purposes in IDX: Adding the shapes of the pins to the component can contribute to a more detailed 3D component model where this is required, but more importantly pins can be used to help correctly orientate detailed component models which may be referenced in an external library by the MCAD system.

By defining the location of the Primary Pin or “Pin 1”, and one or more other pins, the MCAD system should be able to correctly orientate a detailed component model which it would be unable to orientate by shape alone.

How to represent in Component Pins in IDX

Prior to IDXv4.0, Component Pins could be represented by a *FunctionalItemShape* object with a *ShapeElementType* type of *ComponentTerminal*. IDXv4.0 simplifies the process of defining Component Pins by adding optional *PackagePin* objects to the *EDMDItem* object.

Zero or more *PackagePins* can be defined, each with their own insertion point and shape. Where one or more *PackagePins* is defined in an *EDMDItem*, the Primary Pin (or “Pin 1”) should be identified by setting its primary attribute to true. All other pins in the *EDMDItem* should have a primary attribute of false.

XML-Example

Below is an example of how to define two pins on a component. The text in red shows the pin definitions:

```
<foundation:Item id="SYMBOL_2">
      <pdm:ItemType>single</pdm:ItemType>
      <pdm:Identifier Persistant="false">
        <foundation:SystemScope>SYSTEM_1</foundation:SystemScope>
        <foundation:Number>PCB_5</foundation:Number>
        <foundation:Version>1</foundation:Version>
        <foundation:Revision>0</foundation:Revision>
        <foundation:Sequence>0</foundation:Sequence>
      </pdm:Identifier>

      <pdm:PackageName Persistant="false">
        <foundation:SystemScope>SYSTEM_1</foundation:SystemScope>
        <foundation:ObjectName>PLCC28</foundation:ObjectName>
      </pdm:PackageName>

      <pdm:PackagePin pin_number="A1" primary=”true”>
         <d2:Point>POINT_78</d2:Point>
         <pdm:Shape>PIN_SHP_2</pdm:Shape>
      </pdm:PackagePin>

      <pdm:PackagePin pin_number="A2" primary=”false”>
         <d2:Point>POINT_79</d2:Point>
         <pdm:Shape>PIN_SHP_2</pdm:Shape>
      </pdm:PackagePin>

      <foundation:UserProperty xsi:type="property:EDMDUserSimpleProperty">
        <property:Key Persistant="false">
          <foundation:SystemScope>SYSTEM_1</foundation:SystemScope>
          <foundation:ObjectName>PARTNUM</foundation:ObjectName>
        </property:Key>
        <property:Value>100E016</property:Value>
      </foundation:UserProperty>

      <pdm:Shape>COMP_2</pdm:Shape>
 </foundation:Item>
```

6.2.1.6 How to define component thermal properties

Introduction

To describe thermal characteristics of components in IDX several properties are supported. The committee has agreed on the following properties that should be supported by IDX implementation to guarantee correct exchange between different systems.

Supported features are the operating- and maximum power rating, thermal conductivity and the junction to board or case thermal resistance. Property values should always be expressed in the specified units and be numeric only. The Units should not be included in the property value. If for example a maximum power of 500mW is written, the property value of 'POWER_MAX' would be written as '0.5' rather than '0.5W' or '500mW'.

How to define thermal properties

These are standard custom properties for thermal data and all values must be in this unit.

Property	Unit	Description
POWER_OPR	Watts (W)	Operating power rating
POWER_MAX	Watts (W)	Maximum power rating
THERM_COND	Watt/Meter C (W/m-C)	Thermal conductivity
THETA_JB	Celsius/Watt (deg C/W)	Junction to board thermal resistance
THETA_JC	Celsius/Watt (deg C/W)	Junction to case thermal resistance

XML-Example

Below is an example of how to define thermal properties of a component. The text in red shows the definition of the operating power rating for the component. The other properties are defined accordingly.

```
<foundation:Item id="CMPITEM209" IsAttributeChanged="false">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>ELECTRIC COMPONENT
       GEOMETRY</foundation:Description>

      <foundation:UserProperty xsi:type="property:EDMDUserSimpleProperty"
       IsChanged="false" IsNew="false">
        <property:Key IsAttributeChanged="false" Persistant="false"
          IsOriginator="false">
          <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
          <foundation:ObjectName>POWER_OPR</foundation:ObjectName>
        </property:Key>
        <property:Value>1234</property:Value>
      </foundation:UserProperty>

      <pdm:ItemType>single</pdm:ItemType>
      <pdm:Identifier IsAttributeChanged="false" Persistant="false"
       IsOriginator="false">
        <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
        <foundation:Number>RBGEOM</foundation:Number>
        <foundation:Version>1</foundation:Version>
        <foundation:Revision>0</foundation:Revision>
        <foundation:Sequence>0</foundation:Sequence>
      </pdm:Identifier>
      <pdm:PackageName IsAttributeChanged="false" Persistant="false"
       IsOriginator="false">
        <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
```

```
        <foundation:ObjectName>cc1210</foundation:ObjectName>
      </pdm:PackageName>
      <pdm:Shape>COMPONENT3</pdm:Shape>
      <pdm:BaseLine>
        <property:Value>true</property:Value>
      </pdm:BaseLine>
</foundation:Item>
```

6.2.1.7 How to define component value and tolerance properties

Introduction

To describe value and tolerance characteristics of components in IDX, user properties will be used on *EDMDItem* of type single which represents the component.

Supported features are:

- Tolerance (for all component types)
- Resistance (for resistors)
- Capacitance (for capacitors)
- Inductance (for inductors)

How to define value and tolerance of a component

All values of user properties must be a decimal number, without a unit. Unit is from SI and is implied based on the name of the user property.

Property	Unit	Description
RESISTANCE	Ohm (Ω)	Component resistance
CAPACITANCE	Farad (F)	Component capacitance
INDUCTANCE	Henry (H)	Component inductance
TOLERANCE	Percent (%)	Component tolerance

XML-Example

Below is an example of how to define capacitance and tolerance of a component. The text in red shows the definition of the capacitance and tolerance for the component. The other properties are defined accordingly.

```
<foundation:Item id="CMPITEM209" IsAttributeChanged="false">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>ELECTRIC COMPONENT
       GEOMETRY</foundation:Description>

      <foundation:UserProperty xsi:type="property:EDMDUserSimpleProperty"
       IsChanged="false" IsNew="false">
        <property:Key IsAttributeChanged="false" Persistant="false"
          IsOriginator="false">
          <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
          <foundation:ObjectName>CAPACITANCE</foundation:ObjectName>
        </property:Key>
        <property:Value>0.000047</property:Value>
      </foundation:UserProperty>

      <foundation:UserProperty xsi:type="property:EDMDUserSimpleProperty"
       IsChanged="false" IsNew="false">
        <property:Key IsAttributeChanged="false" Persistant="false"
          IsOriginator="false">
          <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
          <foundation:ObjectName>TOLERANCE</foundation:ObjectName>
        </property:Key>
```

```
            <property:Value>10</property:Value>
          </foundation:UserProperty>
          <pdm:ItemType>single</pdm:ItemType>
          <pdm:Identifier IsAttributeChanged="false" Persistant="false"
           IsOriginator="false">
            <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
            <foundation:Number>RBGEOM</foundation:Number>
            <foundation:Version>1</foundation:Version>
            <foundation:Revision>0</foundation:Revision>
            <foundation:Sequence>0</foundation:Sequence>
          </pdm:Identifier>
          <pdm:PackageName IsAttributeChanged="false" Persistant="false"
           IsOriginator="false">
            <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
            <foundation:ObjectName>cc1210</foundation:ObjectName>
          </pdm:PackageName>
          <pdm:Shape>COMPONENT3</pdm:Shape>
          <pdm:BaseLine>
            <property:Value>true</property:Value>
          </pdm:BaseLine>
    </foundation:Item>
```

6.3 How to model Mounting Holes (Plated and non-plated)

Mounting holes can be defined in two ways in IDX; the 'traditional' way used by all versions of IDX and in a simplified form introduced in IDXv4.0.

Mounting holes are described in a similar way to most IDX features: An *EDMDItem* represents the instance of the hole, which references an *EDMDItem* describing the hole itself, which then references an *EDMDInterStratumFeature* object.

EDMDItem of type single of a hole can contain an optional tag *PackageName* which names the padstack of the hole. Aim of transfering the name of hole padstack is to assist the importing CAD application to be able to instantiate a hole from its library, and not to always have to create a new hole type just for IDX.

Alternative way to define a padstack of a hole is also used. With it, the padstack is defined through a user property named "PADSTACK" inside the *EDMDItemInstance* tag.

The *EDMDInterStratumFeature* object defines the hole as an object that spans the layers or strata of the board and in turn references the hole geometry which is defined as an *EDMDCurveSet2D*.

An *EDMDInterStratumFeature.FeatureType* of "Cutout" is used to represent non-plated mounting holes and an *EDMDInterStratumFeature.FeatureType* of "PlatedCutout" is used to represent plated mounting holes.

The simplified form of representing a mounting hole in IDXv4.0 allows the user to omit the *EDMDInterStratumFeature* providing they have specified a geometryType attribute of either HOLE_PLATED or HOLE_NON_PLATED. Examples of both methods of describing a hole are given below:

Note that holes do not need to be circular; they can be any closed 2D shape.

6.3.1 Describing a hole in IDX: "Traditional" method

1) *EDMDItem* of itemType assembly: Represents the Hole Instance:

```
<foundation:Item id="ITEM59" IsAttributeChanged="false">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>BOARD ASSEMBLY</foundation:Description>
      <pdm:ItemType>assembly</pdm:ItemType>
      <!-- pdm:Identifier object containing item's ID -->
      <pdm:ItemInstance>
           <pdm:Item>PTHITEM58</pdm:Item>
           <pdm:InstanceName IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
           <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
           <foundation:ObjectName>PTH_28::29</foundation:ObjectName>
        </pdm:InstanceName>
        <pdm:Transformation>        <!—2D Transform with hole's x,y location-->
            <pdm:TransformationType>d2</pdm:TransformationType>
            <pdm:xx>1</pdm:xx>
            <pdm:xy>0</pdm:xy>
            <pdm:yx>0</pdm:yx>
```

```
            <pdm:yy>1</pdm:yy>
            <pdm:tx IsAttributeChanged="false">
            <property:Value>86.36</property:Value>
            </pdm:tx>
            <pdm:ty IsAttributeChanged="false">
            <property:Value>2.54</property:Value>
            </pdm:ty>
        </pdm:Transformation>
<!—Optional way to define a hole padstack through ItemInstance is below-->
        <foundation:UserProperty xsi:type="property:EDMDUserSimpleProperty"
IsChanged="false" IsNew="false">
          <property:Key IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
            <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
            <foundation:ObjectName>PADSTACK</foundation:ObjectName>
          </property:Key>
          <property:Value>HOLE::636195734068975062</property:Value>
        </foundation:UserProperty>
    </pdm:ItemInstance>
    <pdm:AssembleToName>PCB</pdm:AssembleToName>  <!—Optional reference surface-
->
</foundation:Item>
```

2) *EDMDItem* of itemType single: Represents the Hole itself:

```
<foundation:Item id="PTHITEM58" IsAttributeChanged="false">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>Reusable Padstack</foundation:Description>
      <pdm:ItemType>single</pdm:ItemType>
      <!-- pdm:Identifier object containing item's ID -->
      <pdm:Shape>HOLEFEATURE28</pdm:Shape>
      <!-- pdm:PackageName is optional name of padstack when associated with
EDMDItem of type single of a hole -->
      <pdm:PackageName xsi:type="foundation:EDMDName">
        <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
        <foundation:ObjectName>MTH_7.5MM_KEEPOUT8.5MM_NP</foundation:ObjectName>
      </pdm:PackageName>
    </foundation:Item>
```

3) *InterStratumFeature* Object: *InterStratumFeatureType* of *PlatedCutout* specifies that item is a Plated Hole. An *InterStratumFeatureType* of *PlatedCutout* specifies that the item is a Non-plated Hole:

```
<foundation:InterStratumFeature xsi:type="pdm:EDMDInterStratumFeature"
id="HOLEFEATURE28" IsAttributeChanged="false">
      <pdm:ShapeElement>PTHSHAPE30</pdm:ShapeElement>
      <pdm:InterStratumFeatureType>PlatedCutout</pdm:InterStratumFeatureType>
</foundation:InterStratumFeature>
```

4) *ShapeElement* PTHSHAPE30 will describe the shape of the Hole (not shown)

6.3.1.1 Describing a hole in IDX: “Simplified” method

The following XML example shows how to represent a mounting hole using the ‘simplified’ method introduced in IDXv4.0. By specifying the initial item’s geometryType we can omit the *EDMDInterStratumFeature* object:

1) EDMDItem of itemType assembly: Represents the Hole Instance. The geometryType attribute tells us this Item represents a Plated Hole:

```
<foundation:Item id="ITEM59" geometryType=”HOLE_PLATED”
IsAttributeChanged="false">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>BOARD ASSEMBLY</foundation:Description>
      <pdm:ItemType>assembly</pdm:ItemType>
      <!-- pdm:Identifier object containing item's ID -->
      <pdm:ItemInstance>
           <pdm:Item>PTHITEM58</pdm:Item>
           <pdm:InstanceName IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
           <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
           <foundation:ObjectName>PTH_28::29</foundation:ObjectName>
        </pdm:InstanceName>
        <pdm:Transformation>        <!—2D Transform with hole’s x,y location-->
            <pdm:TransformationType>d2</pdm:TransformationType>
            <pdm:xx>1</pdm:xx>
            <pdm:xy>0</pdm:xy>
            <pdm:yx>0</pdm:yx>
            <pdm:yy>1</pdm:yy>
            <pdm:tx IsAttributeChanged="false">
            <property:Value>86.36</property:Value>
            </pdm:tx>
            <pdm:ty IsAttributeChanged="false">
            <property:Value>2.54</property:Value>
            </pdm:ty>
        </pdm:Transformation>
<!—Optional way to define a hole padstack through ItemInstance is below-->
        <foundation:UserProperty xsi:type="property:EDMDUserSimpleProperty"
IsChanged="false" IsNew="false">
          <property:Key IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
            <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
            <foundation:ObjectName>PADSTACK</foundation:ObjectName>
          </property:Key>
```

```
            <property:Value>HOLE::636195734068975062</property:Value>
        </foundation:UserProperty>
    </pdm:ItemInstance>
    <pdm:AssembleToName>PCB</pdm:AssembleToName>  <!—Optional reference surface-
->
</foundation:Item>
```

2) EDMDItem of itemType single: Represents the Hole itself:

```
<foundation:Item id="PTHITEM58" IsAttributeChanged="false">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>Reusable Padstack</foundation:Description>
      <pdm:ItemType>single</pdm:ItemType>
      <!-- pdm:Identifier object containing item's ID -->
      <pdm:Shape> PTHSHAPE30 </pdm:Shape>
      <!-- pdm:PackageName is optional name of padstack when associated with
EDMDItem of type single of a hole -->
      <pdm:PackageName xsi:type="foundation:EDMDName">
        <foundation:SystemScope>ECADSYSTEM</foundation:SystemScope>
        <foundation:ObjectName>MTH 7.5MM KEEPOUT8.5MM NP</foundation:ObjectName>
      </pdm:PackageName>
    </foundation:Item>
```

3) *ShapeElement* PTHSHAPE30 will describe the shape of the Hole (not shown) and can be referenced directly without the need for an intermediate *EDMDInterStratumFeature* object as we already know the item represents a plated hole.

6.3.1.2 Supported Hole types

The following hole types are supported in IDXv4.0:

Hole Type	Traditional Representation ***EDMDInterStratumFeature.FeatureType*** =	Simplified Representation ***EDMDItem.geometryType*** =
Plated	PlatedCutout	HOLE_PLATED
Non-Plated	Cutout	HOLE_NONPLATED
Plated	PlatedMilledCutout*	HOLE_PLATED_MILLED
Non-Plated	MilledCutout*	HOLE_NONPLATED_MILLED

Table 5: Supported hole types in IDXv4.0

6.4 How to model a Milled Cutout (Plated and non-plated)

A “milled cut-out” is a cut out in a board created by a circular mill or drill following a 2D path.

The resulting shape will appear as a wide line with round end caps and joints, where the area of the line shape is understood to be the cut-out. A milled cut-out may span only some of the board layers.

A milled cut-out is described in a very similar way to a hole, but the shape definition is more complex.

Milled cutouts can be defined as plated or non-plated.

How is a milled cut-out represented in IDX?

This should be represented in IDX as follows (see Figure 29):

- An *EDMDInterStratumFeature* of type ”MilledCutout” if the cutout is not plated, or “PartiallyPlatedCutout” if the cutout is plated.
- Connected to an (inverted) *EDMDShapeElement* of type “PartMountingFeature” if the cut-out is not plated, or of type “PartMountingFeature” if the cut-out is plated.
- Connect to an *EDMDCurveSet2D* with upper & lower bounds spanning the Z range that is to be cut away.
- Connected to an open or closed poly line-arc geometry representing the cutter path, having a “Thickness” equal to the drill diameter.

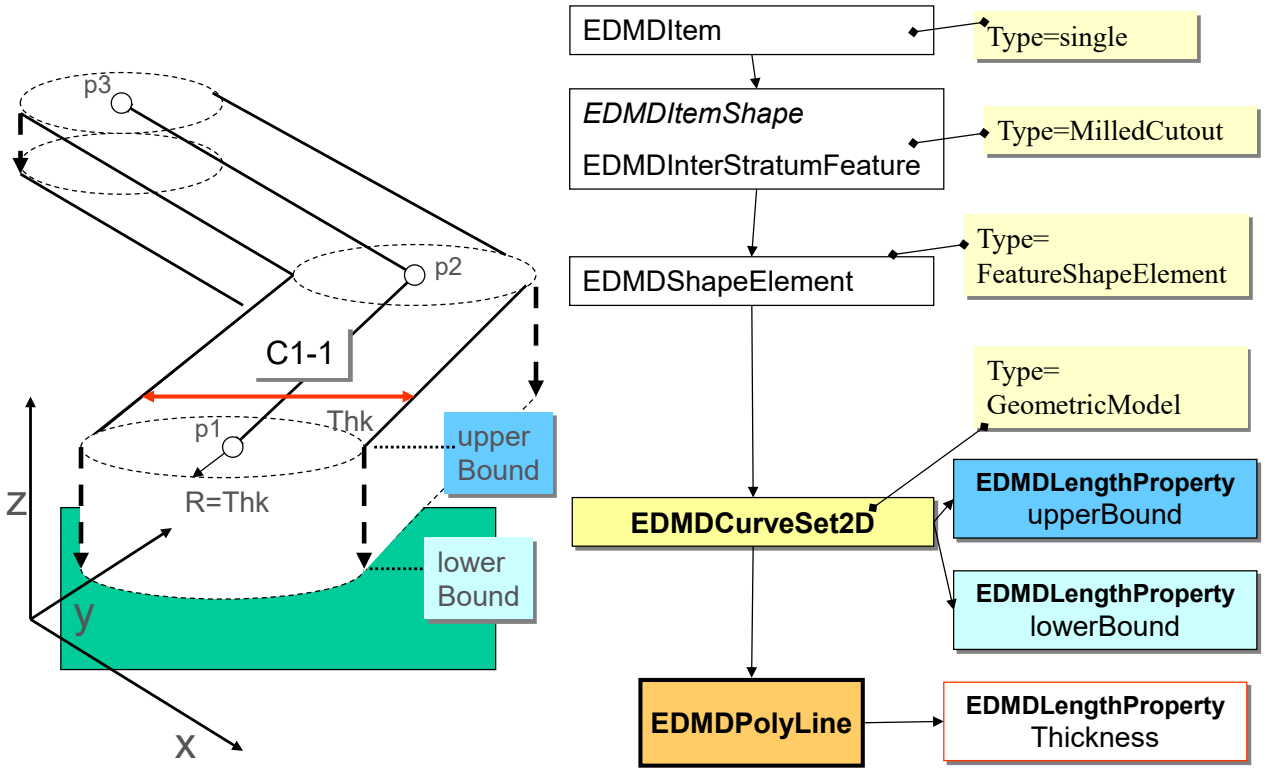


Figure 29: Representation of a "milled cut-out"

Note that XML example files showing the definition of a milled cut-out are available as file:

section5-4-9/baseline_milled_cutout_4.0.idx

6.4.1.1 Describing a milled cut-out in IDX: “Traditional” method

1) *EDMDItem* of itemType assembly: Represents the milled cut-out instance. This is identical to defining any other hole.

```
<foundation:Item id="ITEM59" IsAttributeChanged="false">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>BOARD ASSEMBLY</foundation:Description>
      <pdm:ItemType>assembly</pdm:ItemType>
      <!-- pdm:Identifier object containing item's ID -->
      <pdm:ItemInstance>
           <pdm:Item>PTHITEM58</pdm:Item>
           <pdm:InstanceName    IsAttributeChanged="false"    Persistant="false"
IsOriginator="false">
           <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
           <foundation:ObjectName>PTH_28::29</foundation:ObjectName>
        </pdm:InstanceName>
        <pdm:Transformation>        <!—2D Transform with hole’s x,y location-->
            <pdm:TransformationType>d2</pdm:TransformationType>
            <pdm:xx>1</pdm:xx>
            <pdm:xy>0</pdm:xy>
            <pdm:yx>0</pdm:yx>
            <pdm:yy>1</pdm:yy>
            <pdm:tx IsAttributeChanged="false">
            <property:Value>86.36</property:Value>
            </pdm:tx>
            <pdm:ty IsAttributeChanged="false">
            <property:Value>2.54</property:Value>
            </pdm:ty>
        </pdm:Transformation>
    </pdm:ItemInstance>
    <pdm:AssembleToName>PCB</pdm:AssembleToName>  <!—Optional reference surface-
->
</foundation:Item>
```

2) *EDMDItem* of *itemType* single: Represents the Hole itself:

```
<foundation:Item id="PTHITEM58" IsAttributeChanged="false">
      <foundation:Name> MILLEDCUTOUT </foundation:Name>
      <foundation:Description> MILLED CUTOUT EXAMPLE </foundation:Description>
      <pdm:ItemType>single</pdm:ItemType>
```

```
      <!-- pdm:Identifier object containing item's ID -->
      <pdm:Shape> INTERSTRATUM19</pdm:Shape>
    </foundation:Item>
```

3) *InterStratumFeature* Object: *InterStratumFeatureType* of *MilledCutout* specifies that item is a Milled Cutout. Note the different values of *InterStratumFeatureType* for plated or non-plated cutouts.

```
<foundation:InterStratumFeature xsi:type="pdm:EDMDInterStratumFeature"
id="INTERSTRATUM19">
    <pdm:ShapeElement>SHAPE18</pdm:ShapeElement>
    <pdm:InterStratumFeatureType>MilledCutout</pdm:InterStratumFeatureType>  <!—
Plated = PartiallyPlatedCutout -->
    <pdm:Stratum>STRATUMBRD</pdm:Stratum>
</foundation:InterStratumFeature>
```

4) *ShapeElement* SHAPE18 specifies its defining shape of CURVESET17 and indicates it is inverted (e.g. a cut). Note the different values of *ShapeElementType* for plated or non-plated cutouts.

```
<foundation:ShapeElement xsi:type="pdm:EDMDShapeElement" id="SHAPE18">
  <pdm:ShapeElementType>FeatureShapeElement</pdm:ShapeElementType>  <!—Plated =
PartMountingFeature -->
  <pdm:Inverted>true</pdm:Inverted>
  <pdm:DefiningShape>CURVESET17</pdm:DefiningShape>
</foundation:ShapeElement>
```

5) The Curveset represents the absolute upper and lower bounds of the cut in the z axis of between z=1.15 and z=1.55, where z=0 is the bottom component mounting surface of the board:

```
<foundation:CurveSet2d xsi:type="d2:EDMDCurveSet2d" id="CURVESET17">
     <pdm:ShapeDescriptionType>GeometricModel</pdm:ShapeDescriptionType>
     <d2:LowerBound xsi:type="property:EDMDLengthProperty">
          <property:Value>1.15</property:Value>
      </d2:LowerBound>
      <d2:UpperBound xsi:type="property:EDMDLengthProperty">
          <property:Value>1.55</property:Value>
      </d2:UpperBound>

<d2:DetailedGeometricModelElement>POLYLN16</d2:DetailedGeometricModelElement>
</foundation:CurveSet2d>
```

6) Finally, a Polyline object defines the cutter diameter and the 2D path of the milled cut:

```
<foundation:PolyLine xsi:type="d2:EDMDPolyLine" id="POLYLN16">
     <d2:Thickness xsi:type="property:EDMDLengthProperty">
         <property:Value>5</property:Value>
      </d2:Thickness>
       <d2:Point>PT12</d2:Point>
       <d2:Point>PT13</d2:Point>
       <d2:Point>PT14</d2:Point>
       <d2:Point>PT15</d2:Point>
</foundation:PolyLine>
```

6.4.1.2 Describing a milled cut-out in IDX: “Simplified” method

The description of a milled cut-out can be simplified slightly using the IDXv 4.0 geometryType attribute. Adding a geometryType of HOLE_NONPLATED_MILLED to the top-level *EDMDItem* negates the need for an *InterStratumFeatureType* of type *MilledCutout*.

For example, if the milled hole example described above could be summarised as:

```
EDMDItem(assembly) ITEM59
      EDMDItem(single) PTHITEM58
            EDMDInterStratumFeature INTERSTRATUM19
                  EDMDShapeElement SHAPE18
                        EDMDCurveSet2d CURVESET17
                              EDMDPolyLine POLYLN16
```

Then by adding an attribute geometryType of value HOLE_NONPLATED_MILLED to ITEM59, PTHITEM58 could reference SHAPE18 directly.

E.g.:

```
EDMDItem(assembly) ITEM59 geometryType=HOLE_NONPLATED_MILLED OR
HOLE_PLATED_MILLED
      EDMDItem(single) PTHITEM58
            EDMDShapeElement SHAPE18
                  EDMDCurveSet2d CURVESET17
                        EDMDPolyLine POLYLN16
```

6.5 How to model Keep-Out and Keep-In areas

“Keepouts” or “Keep-out Areas” are areas on or in the board that either the mechanical or electrical engineer has indicated should be kept free of geometry of a specified type. For example, a ‘Placement Keepout’ or ‘Component Keepout’ is an area where components should not be placed. This is usually because they would physically interfere with another item such as the enclosure of the board.

“Keepins” or “Keep-in Areas” are exactly the opposite. They indicate an area on the board where items of a certain type should be placed or kept within.

Keepout and keepin areas can be defined in two ways in IDX; the ‘traditional’ way used by all versions of IDX and in a simplified form introduced in IDXv4.0.

Both keepouts and keepins can either be defined as 2D areas or given an optional height. Where a keepin has a finite height, it indicates that geometry of the specified type within that area should not exceed the specified height. E.g. a Component Keepin with a height of 20mm would indicate that components within that area should not exceed a height of 20mm.

The geometry of a keepout/keepin is described by an *EDMDCurveSet2d* object. The curveset2d’s *LowerBound* property represents the height above the abstract *EDMDItem* referenced in attribute *AssembleToName* that the keepout region begins (E.g. *AssembleToName* = “TOP”, “BOTTOM”, etc. see section 4.1). The *UpperBound* property represents the top height of the keepout relative to the *EDMDItem* item.

If there is no *AssembleToName* property present in the Item, then the keepout is defined relative to the global coordinate system (e.g. the heights are relative to z=0).

Placement keep outs may extend to “infinity” in one or two directions (see Figure 511 below):

- Undefined *LowerBound* specifies unbound bottom (extends in the direction negative infinitely)
- Undefined *UpperBound* specifies unbound top (extends in the direction positive infinitely)

The referenced abstract item also specifies the mathematical orientation of “positive infinity”. That means “positive infinity” relative to “BOTTOM” item is “negative infinity” relative to “TOP” item.

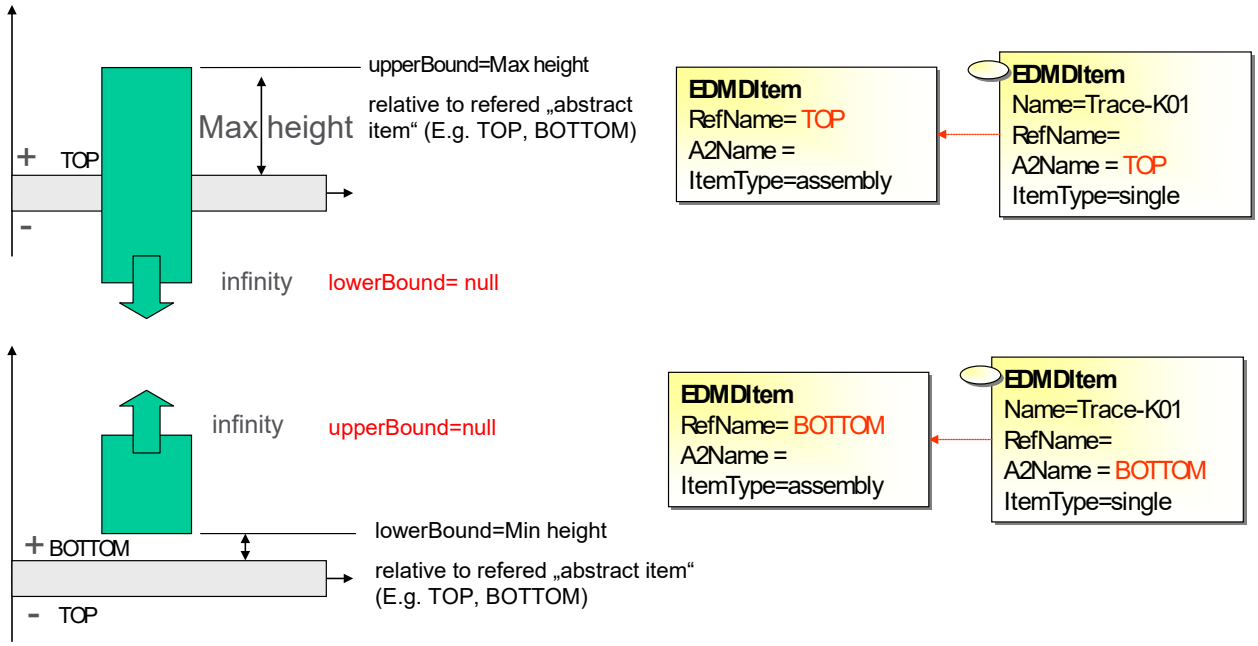


Figure 30: Interpretation of unbound lower/upper bound of keep-out

6.5.1.1 Describing a Keepout in IDX: “Traditional” method

The following XML example shows how to represent a keepout/keepin using the ‘traditional’ method. In this case we are describing a Routing Keepout Area.

1) *EDMDItem* of *itemType=assembly*: Represents the keepout Instance:

```
<foundation:Item id="ITEM191" IsAttributeChanged="false">
      <foundation:Name>Keepout Instance</foundation:Name>
      <foundation:Description>Description of Keepout
Instance</foundation:Description>
      <pdm:ItemType>assembly</pdm:ItemType>
      <!—pdm:Identifier object containing item’s ID -->
      <pdm:ItemInstance>
            <pdm:Item>RKOITEM190</pdm:Item> <!-- This name of the Item
representing the Keepout (below) -->
            <!-- Optional User Properties -->
            <pdm:InstanceName IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
                  <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
                  <foundation:ObjectName>Route_Keepout 1-
All</foundation:ObjectName>    <!-- The name of the Instance -->
             </pdm:InstanceName>
      </pdm:ItemInstance>
      <pdm:AssembleToName>TOP</pdm:AssembleToName>  <!-- Optional surface to
position this keepout relative to -->
 </foundation:Item>
```

2) *EDMDItem* of *itemType=single*: Represents the Keepout itself:

```
<foundation:Item id="RKOITEM190" IsAttributeChanged="false">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>ROUTE OBSTRUCT GEOMETRY</foundation:Description>
      <pdm:ItemType>single</pdm:ItemType>
      <!—pdm:Identifier object containing item’s ID -->
      <pdm:Shape>KEEPOUT1</pdm:Shape>     <!-- By referencing a KeepOut object
we're indicating this is a Keepout -->
    </foundation:Item>
```

3) *EDMDKeepOut* object. Tells us the object is a Keepout and of what type. To represent a Keepin rather than a Keepout we’d use a foundation:KeepIn object here rather than a foundation:KeepOut object:

```
<foundation:KeepOut id="KEEPOUT1" IsAttributeChanged="false">
      <foundation:Name>Routing Keep-Out Area 1</foundation:Name>
```

```
      <pdm:ShapeElement>RKOSHAPE96</pdm:ShapeElement>
      <pdm:Purpose>Route</pdm:Purpose> <!-- This tells us it's a Routing Keepout
-->
</foundation:KeepOut>
```

4) *EDMDShapeElement* RKOSHAPE will describe the shape of the Keepout (not shown)

6.5.1.2 Describing a Keepout in IDX: "Simplified" method

The following XML example shows how to represent a keepout using the 'simplified' method introduced in IDXv4.0. By specifying the initial item's *geometryType* we can omit the *KeepIn* or *KeepOut* object:

1) *EDMDItem* of *itemType=assembly*: Represents the keepout Instance. This time its *geometryType* is specified as KEEPOUT_AREA_ROUTE telling us immediately the object is a Routing Keepout.

```
<foundation:Item id="ITEM191" geometryType="KEEPOUT_AREA_ROUTE"
IsAttributeChanged="false">
      <foundation:Name>Keepout Instance</foundation:Name>
      <foundation:Description>Description of Keepout
Instance</foundation:Description>
      <pdm:ItemType>assembly</pdm:ItemType>
      <!—pdm:Identifier object containing item's ID -->
      <pdm:ItemInstance>
            <pdm:Item>RKOITEM190</pdm:Item>          <!-- This name of the Item
representing the Keepout (below) -->
            <!-- Optional User Properties -->
            <pdm:InstanceName IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
                  <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
                  <foundation:ObjectName>Route_Keepout 1-
All</foundation:ObjectName>    <!-- The name of the Instance -->
             </pdm:InstanceName>
      </pdm:ItemInstance>
      <pdm:AssembleToName>TOP</pdm:AssembleToName>  <!-- Optional surface to
position this keepout relative to -->
 </foundation:Item>
```

2) *EDMDItem* of itemType=single: Represents the Keepout itself. No geometryType is specified as the IDX reader knows this is the item referenced by the item of type KEEPOUT_AREA_ROUTE above.

```
<foundation:Item id="RKOITEM190" IsAttributeChanged="false">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>ROUTE OBSTRUCT GEOMETRY</foundation:Description>
      <pdm:ItemType>single</pdm:ItemType>
      <!—pdm:Identifier object containing item's ID -->
      <pdm:Shape> RKOSHAPE </pdm:Shape>  <!—We already know we have a keepout,
so just reference the shape -->
</foundation:Item>
```

3) *EDMDShapeElement* RKOSHAPE will describe the shape of the Keepout (not shown)

Using the IDXv4.0 simplified representation we omit the need for the *EDMDKeepOut* object. As we already know our item is a keepout we can just reference its shape directly, saving complexity and reducing the size of the file.

6.5.1.3 Supported Keepout/Keepin types

The following keepout types are supported in IDXv4.0:

Keepout Type	Traditional Representation ***EDMDKeepOut.Purpose* =**	Simplified Representation ***EDMDItem.geometryType* =**
Routing	Route	KEEPOUT_AREA_ROUTE
Via	Via	KEEPOUT_AREA_VIA
Thermal	Thermal	KEEPOUT_AREA_THERMAL
Component	ComponentPlacement	KEEPOUT_AREA_COMPONENT
Test Point	TestPoint	KEEPOUT_AREA_TESTPOINT
Other	Other	KEEPOUT_AREA_OTHER

Table 6: Supported keepout types from IDXv4.0

The following keepin types are supported in EDMD 4.0:

Keepin Type	Traditional Representation ***EDMDKeepIn.Purpose* =**	Simplified Representation ***EDMDItem.geometryType* =**
Routing	Route	KEEPIN_AREA_ROUTE
Via	Via	KEEPIN_AREA_VIA
Thermal	Thermal	KEEPIN_AREA_THERMAL
Component	ComponentPlacement	KEEPIN_AREA_COMPONENT
Test Point	TestPoint	KEEPIN_AREA_TESTPOINT
Other	Other	KEEPIn_AREA_OTHER

Table 7: Supported keepin types from IDXv4.0

6.6 How to model Placement Group areas

A placement group area is a closed shape that represents a region on the board that contains related components. Although very similar to a Component Keep-In area, it differs as it does not require placement inside the area. It is an indication to the ECAD or MCAD user that a group of components is related in some way.

A Placement Group area is normally named.

6.6.1.1 Describing a Placement Group area in IDX: “Traditional” method

The following XML example shows how to represent a Placement Group area using the ‘traditional’ method:

1) *EDMDItem* of *itemType=assembly*: Represents the Placement Group Instance:

```
<foundation:Item id="ITEM191" IsAttributeChanged="false">
      <foundation:Name>Keepout Instance</foundation:Name>
      <foundation:Description>Description of Placement Group
Instance</foundation:Description>
      <pdm:ItemType>assembly</pdm:ItemType>
      <!—pdm:Identifier object containing item’s ID -->
      <pdm:ItemInstance>
            <pdm:Item>PGAITEM190</pdm:Item>          <!-- This name of the Item
representing the Group (below) -->
            <!-- Optional User Properties -->
            <pdm:InstanceName IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
                  <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
                  <foundation:ObjectName>Place_Group_1l</foundation:ObjectName>
       <!-- The name of the Instance -->
             </pdm:InstanceName>
      </pdm:ItemInstance>
      <pdm:AssembleToName>TOP</pdm:AssembleToName>  <!-- Optional surface to
position this area relative to -->
 </foundation:Item>
```

2) *EDMDItem* of *itemType=single*: Represents the Placement Group itself:

```
<foundation:Item id=" PGAITEM190" IsAttributeChanged="false">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>Placement Group Area 4</foundation:Description>
      <pdm:ItemType>single</pdm:ItemType>
      <!—pdm:Identifier object containing item’s ID -->
      <pdm:Shape> PLACEGROUPAREA_260</pdm:Shape>     <!-- Referencing a
EDMDFunctionalItemShapeobject -->
    </foundation:Item>
```

3) *FunctionalItemShape* object. *FunctionalItemShapeType* tells us the object is a Placement Group Area:

```
<foundation:FunctionalItemShape id="PLACEGROUPAREA_260">
          <pdm:Name>GroupAreaName</pdm:Name>
          <pdm:ShapeElement>SHAPEELEMENT_267</pdm:ShapeElement>

<pdm:FunctionalItemShapeType>PlacementGroupArea</pdm:FunctionalItemShapeType>
</foundation:FunctionalItemShape>
```

4) *EDMDShapeElement* SHAPEELEMENT_267 will describe the shape of the Area (not shown)

6.6.1.2 Describing a Placement Group area in IDX: “Simplified” method

The description of a placement group can be simplified slightly using the IDXv4.0 *geometryType* attribute. Adding a geometryType of PLACEMENT_GROUP_AREA to the top-level *EDMDItem* negates the need for an *EDMDFunctionalItemShape* of type *PlacementGroupArea*.

For example, if the example described above could be summarised as:

```
EDMDItem(assembly) ITEM191
      EDMDItem(single) PGAITEM190
            FunctionalItemShape PLACEGROUPAREA_260
                  EDMDShapeElement SHAPEELEMENT_267 etc.
```

Then by adding an attribute geometryType of value PLACEMENT_GROUP_AREA to ITEM191, PGAITEM190 could reference SHAPEELEMENT_267 directly.

E.g.:

```
EDMDItem(assembly) ITEM191 geometryType= PLACEMENT_GROUP_AREA
      EDMDItem(single) PGAITEM190
            EDMDShapeElement SHAPEELEMENT_267 etc.
```

In the case of the simplified representation, the Placement Group area name is defined as the *EDMDItem(single).Name* rather than *FunctionalItemShape.Name* which is no longer present.

6.7 How to model “Other Outlines” or User areas

An “Other Outline” or User Area is a catch-all area type that allows the user to defined named regions that do not have a predefined purpose. It is defined as a closed shape that represents a region on the board with some function which is neither strictly electrical nor mechanical, nor is it any kind of keep-in or keep-out area.

For example, an MCAD engineer could suggest a position of a logo on a PCB by creating an “Other Outline” named 'LOGO POSITION ON TOP'. The ECAD engineer would then know to place the logo at this position represented by the Outline.

The definition of an Other Outline is very similar to the definition of a Placement Group Area. The only difference is the *FunctionalItemShapeType* in the *EDMDFunctionalItemShape* object which is of type “UserArea” rather than “*PlacementGroupArea*”.

The Other Outline type also has an optional element “InstanceUserAreaLayerName” on the *ItemInstance* of the top-level Item as shown in the example XML below. This can be used by ECAD systems to map a User Area/Other Outline to a specific design layer in the ECAD system if required. For example, a User Area called ‘Logo’ could be mapped to ECAD design layer called ETCH_TOP as shown

If required, additional optional User Properties can also be defined in the assembly level *EDMDItem* to convey more information to the ECAD or MCAD system.

6.7.1.1 Describing an Other Outline in IDX: “Traditional” method

The following XML example shows how to represent an Other Outline using the ‘traditional’ method.

1) *EDMDItem* of *itemType=assembly*: Represents the Outline Instance:

```
<foundation:Item id=" ITEM24" IsAttributeChanged="false">
      <foundation:Name>LOGO</foundation:Name>
      <foundation:Description>NEW USER AREA FOR LOGO</foundation:Description>
      <pdm:ItemType>assembly</pdm:ItemType>
      <!—pdm:Identifier object containing item’s ID -->
      <pdm:ItemInstance>
            <pdm:Item>ITEM22</pdm:Item>        <!-- This name of the Item
representing the Outline (below) -->
            <!-- Optional User Properties -->
            <pdm:InstanceName IsAttributeChanged="false" Persistant="false"
IsOriginator="false">
                  <foundation:SystemScope>MCADSYSTEM</foundation:SystemScope>
                  <foundation:ObjectName> MECODES::23</foundation:ObjectName>
       <!-- The name of the Instance -->
             </pdm:InstanceName>
             <pdm:InstanceUserAreaLayerName Persistant="false">         <!—
Optional UserAreaLayerName (see above) -->
                  <foundation:SystemScope>SYSTEM_1</foundation:SystemScope>
                  <foundation:ObjectName>ETCH_TOP</foundation:ObjectName>
            </pdm: InstanceUserAreaLayerName >
      </pdm:ItemInstance>
```

```
      <pdm:AssembleToName>TOP</pdm:AssembleToName>  <!-- Optional surface to
position this area relative to -->
 </foundation:Item>
```

2) *EDMDItem* of *itemType=single*: Represents the Outline itself:

```
<foundation:Item id=" ITEM22" IsAttributeChanged="false">
      <foundation:Name>PCB</foundation:Name>
      <foundation:Description>NEW OTHER OUTLINE</foundation:Description>
      <pdm:ItemType>single</pdm:ItemType>
      <!—pdm:Identifier object containing item's ID -->
      <pdm:Shape>FUNCTIONALITEMSHAPE21</pdm:Shape>   <!-- Referencing a
EDMDFunctionalItemShapeobject -->
    </foundation:Item>
```

3) *FunctionalItemShape* object. *FunctionalItemShapeType* tells us the object is an Other Outline:

```
<foundation:FunctionalItemShape id="FUNCTIONALITEMSHAPE21">
          <pdm:Name>LOGO AREA ON TOP</pdm:Name>
          <pdm:ShapeElement>SHAPEELEMENT_267</pdm:ShapeElement>
          <pdm:FunctionalItemShapeType>UserArea</pdm:FunctionalItemShapeType>
</foundation:FunctionalItemShape>
```

4) *EDMDShapeElement* FUNCTIONALITEMSHAPE21will describe the shape of the Area (not shown).

6.7.1.2 Describing an Other Outline in IDX: "Simplified" method

The description of a placement group can be simplified slightly using the IDXv4.0 *geometryType* attribute. Adding a geometryType of OTHER_OUTLINE to the top-level *EDMDItem* negates the need for an *EDMDFunctionalItemShape* of type *UserArea*.

For example, if the example described above could be summarised as:

```
EDMDItem(assembly) ITEM24
      EDMDItem(single) ITEM22
            FunctionalItemShape FUNCTIONALITEMSHAPE21
                  EDMDShapeElement SHAPE20 etc.
```

Then by adding an attribute geometryType of value OTHER_OUTLINE to ITEM24, ITEM22 could reference SHAPE20 directly.

E.g.:

```
EDMDItem(assembly) ITEM24 geometryType= PLACEMENT_GROUP_AREA
      EDMDItem(single) ITEM22
            EDMDShapeElement SHAPE20 etc.
```

In the case of the simplified representation, the Other Outline area name is defined as the *EDMDItem(single).Name* rather than *FunctionalItemShape.Name* which is no longer present.

6.8 How to represent Traces, Copper areas and Silkscreen

IDX also covers the capability to transfer information regarding traces and silkscreen information from the ECAD system to the MCAD system. However, this information is currently treated as non-modifiable in the MCAD system.

The IDX schema does not support declaring items as “read only” .Therefore the IDX schema itself does not limit collaboration on traces as “read only”. The ECAD/MCAD systems implementing the IDX collaboration need to handle this by convention.

The complete wiring of a layer might be described by one item (or could be split into several items). The modeling of traces requires the following elements:

- *EDMDPolyline/EDMDPoint/EDMDCircle/EDMDArc* to define the geometry of wire (trace, pad).
- *EDMDCurveset2d* to group the traces and provide a lower/upper bound for height/thickness
- *EDMDShapeElement* to encapsulate the *EDMDCurveset2d*
- *StratumTechnology* to define Technology Type and *LayerPurpose*
- Stratum to encapsulate the Shape and *StratumTechnology*.

Figure 48 illustrates the IDX representation for traces based on an example. The wiring is described by three polylines, one line and one arc. The name of IDX elements are for illustration only. There is no naming convention to be followed.

The *LayerPurpose* can be used to differentiate between traces, pads, masks, silkscreen and other generic shapes or drawing layers as follows:

- *PowerOrGround* (Copper planes)
- *OtherSignal* (Traces)
- SolderMask (Soldermasks)
- and *SilkScreen* (generic shapes)

The *TechnologyType* can be either “Design” or “Documentation”. For physical layers of the board, the *TechnologyType* should be set to “Design”.

The *SimpleUserProperty* called “SIDE” is used to identify the location of the layer, either on TOP of the board, the BOTTOM of the board, or INNER for inside the board.

(The *StratumTechnology* “LayerPositionType” and the Stratum “StratumSurfaceDesignation” and “StratumType” enums can be deprecated.)

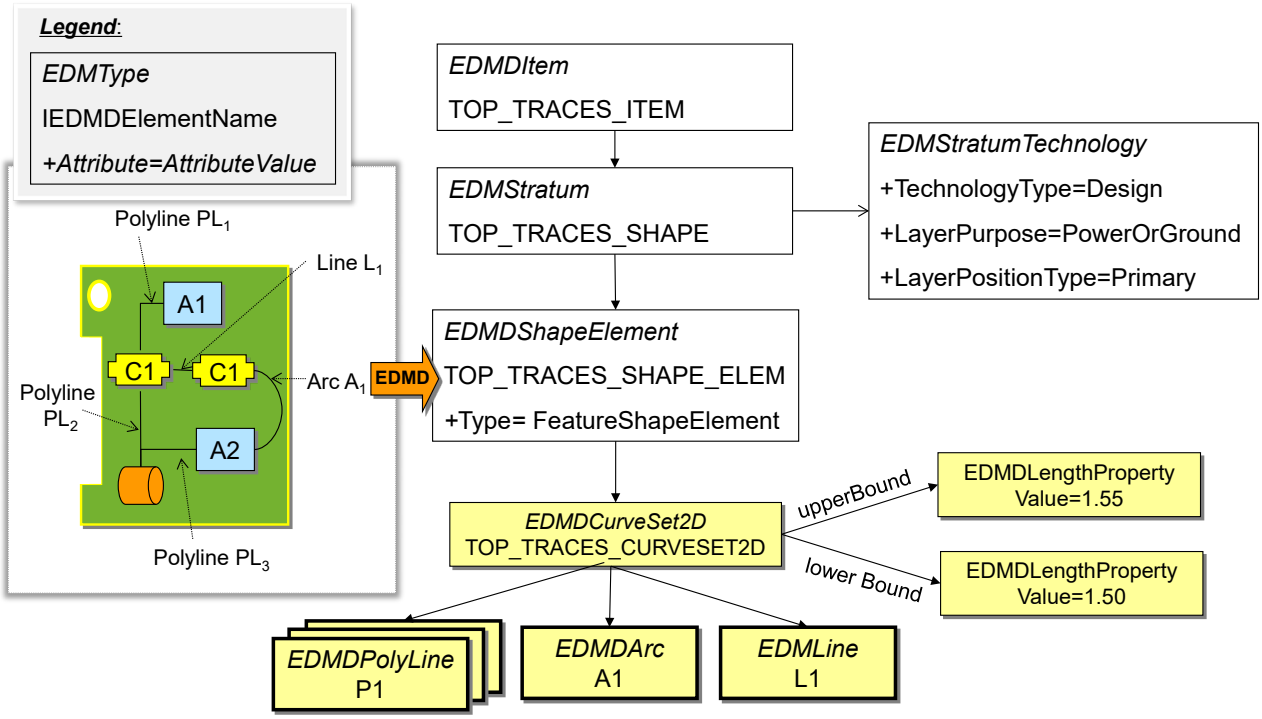


Figure 31: Example for representation of traces in IDX

Collaboration on copper areas is supported by IDX. Therefore, copper areas are described by a second item. The shape of this type of item consists of closed polyline-arcs with cutouts, like the board.

XML-Example Traces non-collaborative:

```
    <foundation:CartesianPoint id="POINT_1">
      <d2:X>
        <property:Value>1650</property:Value>
      </d2:X>
      <d2:Y>
        <property:Value>387</property:Value>
      </d2:Y>
    </foundation:CartesianPoint>
    <foundation:CartesianPoint id="POINT_2">
      <d2:X>
        <property:Value>1650</property:Value>
      </d2:X>
      <d2:Y>
        <property:Value>475</property:Value>
      </d2:Y>
    </foundation:CartesianPoint>
    <foundation:PolyLine id="POLYLINE_1">
      <!- If the thickness is present, the polyline does not represent a closed
polygon.
           The MCAD tool should represent the line with its thickness.
           Otherwise, it is a closed polygon ->
      <d2:Thickness>
        <property:Value>8</property:Value>
      </d2:Thickness>
      <d2:Point>POINT_1</d2:Point>
      <d2:Point>POINT_2</d2:Point>
    </foundation:PolyLine>
    <foundation:CurveSet2d id="CURVESET2D_1">
      <pdm:ShapeDescriptionType>GeometricModel</pdm:ShapeDescriptionType>
```

```
      <d2:LowerBound>
        <property:Value>0</property:Value>
      </d2:LowerBound>
      <d2:UpperBound>
        <property:Value>150</property:Value>
      </d2:UpperBound>
      <d2:DetailedGeometricModelElement>POLYLINE_1</d2:DetailedGeometricModelEle
ment>
    </foundation:CurveSet2d>
    <foundation:ShapeElement id="SHAPEELEMENT_1">
      <foundation:Description>NET="ZEROCROSS"</foundation:Description>
      <pdm:ShapeElementType>FeatureShapeElement</pdm:ShapeElementType>
      <pdm:DefiningShape>CURVESET2D_1</pdm:DefiningShape>
    </foundation:ShapeElement>

    <foundation:StratumTechnology id="stratumTechnology_1">
      <pdm:TechnologyType>Design</pdm:TechnologyType>
      <pdm:LayerPurpose>PowerOrGround</pdm:LayerPurpose>
    </foundation:StratumTechnology>

    <foundation:Stratum id="stratum_1">
      <pdm:ShapeElement>SHAPEELEMENT_1</pdm:ShapeElement>
      <pdm:StratumTechnology>stratumTechnology_1</pdm:StratumTechnology>
    </foundation:Stratum>

    <foundation:Item id="ITEM_1">
      <pdm:ItemType>single</pdm:ItemType>
      <pdm:Identifier Persistant="false">
        <foundation:SystemScope>SYSTEM_1</foundation:SystemScope>
        <foundation:Number>PCB_1</foundation:Number>
        <foundation:Version>1</foundation:Version>
        <foundation:Revision>0</foundation:Revision>
        <foundation:Sequence>0</foundation:Sequence>
      </pdm:Identifier>
      <pdm:Shape>stratum_1</pdm:Shape>
    </foundation:Item>
    <foundation:Item id="ITEM_2">
      <pdm:ItemType>assembly</pdm:ItemType>
      <pdm:Identifier Persistant="false">
        <foundation:SystemScope>SYSTEM_1</foundation:SystemScope>
        <foundation:Number>PCB_2</foundation:Number>
        <foundation:Version>1</foundation:Version>
        <foundation:Revision>0</foundation:Revision>
        <foundation:Sequence>0</foundation:Sequence>
      </pdm:Identifier>
      <pdm:ItemInstance id="ITEMINST_1">
        <foundation:UserProperty xsi:type="property:EDMDUserSimpleProperty">
          <property:Key Persistant="false">
            <foundation:SystemScope>SYSTEM_1</foundation:SystemScope>
            <foundation:ObjectName>SIDE</foundation:ObjectName>
          </property:Key>
          <property:Value>TOP</property:Value>
        </foundation:UserProperty>
        <pdm:InstanceName Persistant="false">
          <foundation:SystemScope>SYSTEM_2</foundation:SystemScope>
          <foundation:ObjectName>ETCH_TOP</foundation:ObjectName>
        </pdm:InstanceName>
        <pdm:Item>ITEM_1</pdm:Item>
      </pdm:ItemInstance>
    </foundation:Item>
```

XML-Example Copper Area:

```
      <foundation:CartesianPoint id="PT10">
        <d2:X>
          <property:Value>19.000000</property:Value>
        </d2:X>
        <d2:Y>
          <property:Value>4.000000</property:Value>
        </d2:Y>
      </foundation:CartesianPoint>
      <foundation:CartesianPoint id="PT5">
        <d2:X>
          <property:Value>6.000000</property:Value>
        </d2:X>
        <d2:Y>
          <property:Value>4.000000</property:Value>
        </d2:Y>
      </foundation:CartesianPoint>
      <foundation:CartesianPoint id="PT6">
        <d2:X>
          <property:Value>6.000000</property:Value>
        </d2:X>
        <d2:Y>
          <property:Value>94.000000</property:Value>
        </d2:Y>
      </foundation:CartesianPoint>
      <foundation:PolyLine id="POLYLN10">
        <d2:Point>PT10</d2:Point>
        <d2:Point>PT5</d2:Point>
      </foundation:PolyLine>
      <foundation:PolyLine id="POLYLN5">
        <d2:Point>PT5</d2:Point>
        <d2:Point>PT6</d2:Point>
      </foundation:PolyLine>
      <foundation:PolyLine id="POLYLN6">
        <d2:Point>PT6</d2:Point>
        <d2:Point>PT10</d2:Point>
      </foundation:PolyLine>
      <foundation:CurveSet2d id="CURVE2">
        <pdm:ShapeDescriptionType>GeometricModel</pdm:ShapeDescriptionType>
        <d2:LowerBound>
          <property:Value>0.000000</property:Value>
        </d2:LowerBound>
        <d2:UpperBound>
          <property:Value>0.000000</property:Value>
        </d2:UpperBound>

<d2:DetailedGeometricModelElement>POLYLN5</d2:DetailedGeometricModelElement>

<d2:DetailedGeometricModelElement>POLYLN6</d2:DetailedGeometricModelElement>

<d2:DetailedGeometricModelElement>POLYLN10</d2:DetailedGeometricModelElement>
      </foundation:CurveSet2d>
      <foundation:ShapeElement id="SHAPE2">
        <pdm:ShapeElementType>FeatureShapeElement</pdm:ShapeElementType>
        <pdm:DefiningShape>CURVE2</pdm:DefiningShape>
      </foundation:ShapeElement>
      <foundation:FunctionalItemShape id="FUNCTITEM1">
        <pdm:ShapeElement>SHAPE2</pdm:ShapeElement>
        <pdm:FunctionalItemShapeType>UserArea</pdm:FunctionalItemShapeType>
      </foundation:FunctionalItemShape>
```

```
      <foundation:Item id="ITEM003">
        <foundation:Name>PCB</foundation:Name>
        <pdm:ItemType>single</pdm:ItemType>
        <pdm:Identifier xsi:type="foundation:EDMDIdentifier">
          <foundation:SystemScope>SYSTEM_1</foundation:SystemScope>
          <foundation:Number></foundation:Number>
          <foundation:Version>1</foundation:Version>
          <foundation:Revision>0</foundation:Revision>
          <foundation:Sequence>0</foundation:Sequence>
        </pdm:Identifier>
        <pdm:Shape>FUNCTITEM1</pdm:Shape>
        <pdm:BaseLine>
          <property:Value>true</property:Value>
        </pdm:BaseLine>
      </foundation:Item>
      <foundation:Item id="ITEM004">
        <foundation:Name>PCB</foundation:Name>
        <pdm:ItemType>assembly</pdm:ItemType>
        <pdm:Identifier xsi:type="foundation:EDMDIdentifier">
          <foundation:SystemScope>SYSTEM_1</foundation:SystemScope>
          <foundation:Number>PCB_CopperEtch</foundation:Number>
          <foundation:Version>1</foundation:Version>
          <foundation:Revision>0</foundation:Revision>
          <foundation:Sequence>0</foundation:Sequence>
        </pdm:Identifier>
        <pdm:ItemInstance id="ITEMINST004"
xsi:type="foundation:EDMDItemInstance">
          <foundation:Name>CopperEtch</foundation:Name>
          <pdm:InstanceName>
            <foundation:SystemScope>SYSTEM_2</foundation:SystemScope>
            <foundation:ObjectName> OtherOutline1</foundation:ObjectName>
          </pdm:InstanceName>
          <foundation:UserProperty xsi:type="property:EDMDUserSimpleProperty">
            <property:Key xsi:type="foundation:EDMDName">
              <foundation:SystemScope>SYSTEM_1</foundation:SystemScope>
              <foundation:ObjectName>SIDE</foundation:ObjectName>
            </property:Key>
            <property:Value>TOP</property:Value>
          </foundation:UserProperty>
          <!- This is a temporary way, in the EDMD 3.0 schema, to identify the
user area
              class.->
          <foundation:UserProperty xsi:type="property:EDMDUserSimpleProperty">
            <property:Key Persistant="false">
              <foundation:SystemScope>SYSTEM_1</foundation:SystemScope>
              <foundation:ObjectName>MAPPED_LAYER</foundation:ObjectName>
            </property:Key>
            <property:Value>ETCH_TOP</property:Value>
          </foundation:UserProperty>
          <!- In EDMD 3.5, the InstanceUserAreaLayerName will
              replace this user property. ->
          <pdm:InstanceUserAreaLayerName Persistant="false">
            <foundation:SystemScope>SYSTEM_1</foundation:SystemScope>
            <foundation:ObjectName>ETCH_TOP</foundation:ObjectName>
          </pdm:InstanceName>
          <pdm:Item>ITEM003</pdm:Item>
        </pdm:ItemInstance>
        <pdm:BaseLine>
          <property:Value>true</property:Value>
        </pdm:BaseLine>
      </foundation:Item>
```

XML example files in attachment (zip):

section5-4-1\baseline_4_0_boardoutline_copperpad.idx

section5-4-1\baseline_4_0_boardoutline_coppertrace.idx

section5-4-1\baseline_4_0_boardoutline_coppervia.idx

6.9 Summary of all Item Types

IDXv4.0 geometryType attribute	EDMDItem ->	EDMDItem ->	(Object varies by Geometry type) ->	EDMDShapeElement
	(Need one of these for each Item instance)	*(Represents the item itself, so only need one)*	*This can be omitted in most cases if top level ITEM has a geometryType attribute*	
FOR ALL ITEMS *All items follow this pattern and have these properties:*	id = ITEM1 ItemType = assembly ItemInstance.Item = ITEM2	id = ITEM2 itemType = single Shape = SHAPEITEM1	id = SHAPEITEM1 ShapeElement = SHAPEELEMENT1	id = SHAPEELEMENT1 DefiningShape = CURVESET1 Inverted = false/true
BOARD_OUTLINE			**EDMDStratum** StratumType = DesignLayerStratum StratumSurfaceDesignation = PrimarySurface	ShapeElementTypeFeature = FeatureShapeElement
Common to all EDMDKeepIn types: KEEPIN_AREA_ROUTING KEEPIN_AREA_COMPONENT KEEPIN_AREA_VIA KEEPIN_AREA_TESTPOINT KEEPIN_AREA_OTHER NOT YET SUPPORTED	AssembleToName = *(name of layer stackup)* - e.g. this keepin on this Layer/spans this Layer Stackup		**EDMDKeepIn** Purpose = Route Purpose = ComponentPlacement Purpose = Via Purpose = TestPoint Purpose = Other Purpose = ValueLimit Purpose = Plane Purpose = Thermal Purpose = GenericClearance Purpose = Shock Purpose = Vibration Purpose = ElectromagneticCompatibility	ShapeElementTypeFeature = FeatureShapeElement
Common to all EDMDKeepOut types: KEEPOUT_AREA_ROUTING KEEPOUT_AREA_COMPONENT	AssembleToName = *(name of layer stackup)* - e.g. this keepin on this Layer/spans this Layer Stackup		**EDMDKeepOut** Purpose = Route Purpose = ComponentPlacement	ShapeElementTypeFeature = FeatureShapeElement

KEEPOUT_AREA_VIA KEEPOUT_AREA_TESTPOINT KEEPOUT_AREA_OTHER *NOT YET SUPPORTED*			Purpose = Via Purpose = TestPoint Purpose = Other Purpose = ValueLimit Purpose = Plane Purpose = Thermal Purpose = GenericClearance Purpose = Shock Purpose = Vibration Purpose = ElectromagneticCompatibility	
Common to all EDMDFunctionalItemShape types:	AssembleToName = *(name of layer stackup)* - e.g. this zone to span these layers		**EDMDFunctionalItemShape**	ShapeElementTypeFeature = FeatureShapeElement
PLACEMENT_GROUP_AREA			FunctionalItemShapeType = PlacementGroupArea Name = *Group Area Name*	
OTHER_OUTLINE	ItemInstance.UserAreaLayerName = (e.g.) "ETCH_TOP" **v4.0 only*		FunctionalItemShapeType = UserArea Name = *User Area/Other Outline Name*	
BOARD_AREA_FLEXIBLE BOARD_AREA_STIFFENER BOARD_AREA_RIGID*			FunctionalItemShapeType = FlexibleArea FunctionalItemShapeType = Stiffener	
Common to all EDMDInterStratumFeature types:	ItemInstance.UserProperty [MHNAME] = Plated Hole 2 ItemInstance.UserProperty [PADSTACK] = PTH_2 ItemInstance.UserProperty [ICMIDX_ATTRIBUTE_HOLE_ASSOCIATED_PART] = BOARD ItemInstance.UserProperty [ICMIDX_ATTRIBUTE_HOLE_TYPE_USER_DEFINED] = 138-NPTH *Cadence		**EDMDInterStratumFeature**	Inverted = true
HOLE_PLATED			InterStratumFeatureType = PlatedCutout	ShapeElementType = DesignLayerStratum/FeatureShapeElement/PartMountingFeature
VIA			InterStratumFeatureType = Via	ShapeElementType = DesignLayerStratum
FILLED_VIA			InterStratumFeatureType = FilledVia	ShapeElementType = DesignLayerStratum

HOLE_NON_PLATED HOLE_PLATED_MILLED ("Partially Plated Cutout") HOLE_NONPLATED_MILLED			InterStratumFeatureType = Cutout InterStratumFeatureType = PartiallyPlatedCutout InterStratumFeatureType = MilledCutout	ShapeElementType = DesignLayerStratum ShapeElementType = PartMountingFeature ShapeElementType = FeatureShapeElement
Common to all EDMDAssemblyComponent types: COMPONENT COMPONENT_MECHANICAL	ItemInstance.Name = *Reference Designator, e.g. "C6"* AssembleToName = *(name of layer stackup)* - Optional Layer reference for location in z axis UserProperty [Placed] = true/false ItemInstance.UserProperty [REFDES] = C16 * Mentor ItemInstance.UserProperty [PARTNUM] = 12333-CAP	PackageName: CC1206	**EDMDAssemblyComponent** AssemblyComponentType = Physical AssemblyComponentType = MechanicalItem	ShapeElementType = FeatureShapeElement
Common to all EDMDStratum types: COPPER_PAD COPPER_TRACE COPPER_AREA SOLDERMASK SILKSCREEN			**EDMDStratum** StratumType = DesignLayerStratum StratumTechnology = EDMDStratumTechnology.LayerPurpose.LandsOnly StratumTechnology = EDMDStratumTechnology.LayerPurpose.OtherSignal StratumTechnology = EDMDStratumTechnology.LayerPurpose.PowerOrGRound StratumTechnology = EDMDStratumTechnology.LayerPurpose.SolderMask StratumTechnology = EDMDStratumTechnology.LayerPurpose.Silkscreen	ShapeElementType = FeatureShapeElement
BEND			**EDMDBend** BendType = EDMDCircularBendType InnerSide = Bottom BendLine = LINE_1 InnerRadius = 600.000 InnerAngle = 90.000	ShapeElementType = NonFeatureShapeElement

Common to all Layers:	ItemInstance.Item = ITEM2 (Refers to an EDMDItem of type assembly representing a Layer) ReferenceName = Inner_Layer_1 *(Will be referenced by Layer's StackUp's ITEM property)* UserProperty [LowerBound] = 0 *(Optional, but must be specified in the Layer or Layer Stackup)* UserProperty [UpperBound] = 36.3 *(Optional, but must be specified in the Layer or Layer Stackup)*
LAYER_SOLDERMASK	UserProperty [LayerType] = SolderMask
LAYER_SOLDERPASTE	UserProperty [LayerType] = SolderPaste
LAYER_SILKSCREEN	UserProperty [LayerType] = SilkScreen
LAYER_GENERIC	UserProperty [LayerType] = GenericLayer
LAYER_GLUE	UserProperty [LayerType] = Glue
LAYER_GLUEMASK	UserProperty [LayerType] = GlueMask
LAYER_PASTEMASK	UserProperty [LayerType] = PasteMask
LAYER_OTHERSIGNAL	UserProperty [LayerType] = OtherSignal
LAYER_LANDSONLY	UserProperty [LayerType] = LandsOnly
LAYER_POWERORGROUND	UserProperty [LayerType] = PowerOrGround
LAYER_EMBEDDED_CAP_DIELECTRIC	UserProperty [LayerType] = EmbeddedPassiveCapacitorDielectric
LAYER_EMBEDDED_RESISTOR	UserProperty [LayerType] = EmbeddedPassiveResistor

LAYER_DIELECTRIC LAYER_STIFFENER	UserProperty [LayerType] = Dielectric UserProperty [LayerType] = Stiffener
LAYER_STACKUP	Id = ITEM1 ItemType = assembly ItemInstance.Item = ITEM2 Item has a pdm:ItemInstance for each Layer in the Stackup with these properties: Instance.Item = ID of the referenced Layer object e.g. ITEM1 Instance.UserProperty [LayerType] = Soldermask, Signal, Dielectric etc… Instance.UserProperty [LowerBound] = 0 *(Optional, but must be specified in the Layer or Layer Stackup)* Instance.UserProperty [UpperBound] = 36.3 *(Optional, but must be specified in the Layer or Layer Stackup)*

**The 'Simple' notation should be used to define this item as it has no equivalent definition using the 'Traditional' notation.*

Table 8: Summary of all Item Types

6.10 Mapping IDF Items to IDX Items

The table below shows the equivalent items to use in the IDX format for items found in the IDF 2.0 and 3.0 formats. Note that IDF does not support more complex entity types such as Layers.

IDF File section	IDXv4.0 *geometryType*	Equivalent IDX Object **(Will not be required in most cases if geometryType is specified)**
.BOARD_OUTLINE	BOARD_OUTLINE	EDMDStratum
.PANEL_OUTLINE	n/a	n/a
.OTHER_OUTLINE	OTHER_OUTLINE	EDMDFunctionalItemShape
.ROUTE_OUTLINE	KEEPIN_AREA_ROUTING	EDMDKeepIn
.PLACE_OUTLINE	KEEPIN_AREA_COMPONENT	EDMDKeepIn
.ROUTE_KEEPOUT	KEEPOUT_AREA_ROUTING	EDMDKeepOut
.VIA_KEEPOUT	KEEPOUT_AREA_VIA	EDMDKeepOut
.PLACE_KEEPOUT	KEEPOUT_AREA_COMPONENT	EDMDKeepOut
.PLACE_REGION	PLACEMENT_GROUP_AREA	EDMDFunctionalItemShape
.DRILLED_HOLES (PTH)	HOLE_PLATED	EDMDInterStratumFeature
.DRILLED_HOLES (NPTH)	HOLE_NON_PLATED	EDMDInterStratumFeature
.PLACEMENT	COMPONENT or COMPONENT_MECHANICAL	EDMDAssemblyComponent
.NOTES	TEXT	EDMDText

Table 9: Mapping of IDF Items to IDX Items

7 Describing Geometry in IDX

This section describes how to describe 2D and 3D geometry in IDX. The objects and methods used to describe geometry are common to all Item types in the format. The shape of a board cutout is described in exactly the same way as the shape of a keep out for example.

Specific information about how to describe each type of specific feature is given in the next section.

7.1 Representing the shape of an item in “2.5 D”

The description of geometry in IDX is limited to the specification of a “translation body” (2D curves and thickness (so called “2.5 D”)). This section illustrates how to describe the shape of an item in IDX.

A shape may consist of several translation bodies. The shape of an item is described by an ItemShape, which comprises a number of different parts:

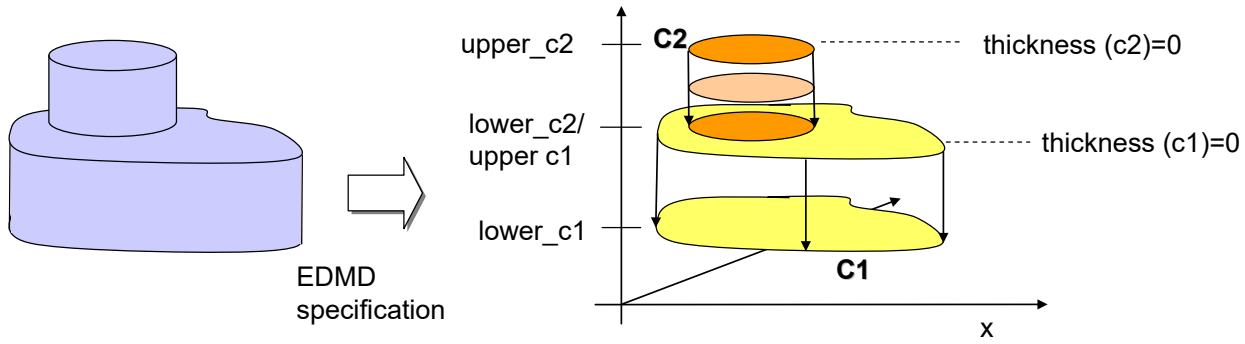


Figure 32: Representation of a shape of an item by combine several “translation bodies” (“OR” operator)

A possible instantiation is shown in Figure 34. The item to be described is a PCB element. Thus, the class *EDMDAssemblyComponent* has been chosen for *ItemShape*. The *ItemShape* is described by one *ShapeElement*. The defining shape of the *ShapeElement* is an *EDMDCurveSet2D* (c12). The *CurveSet2D* represents the composition of two translation bodies. The footprint (base area) of the first body (B1) is described by a spline curve (C1) and the second (B2) by an ellipse (c2). The height of these bodies could be specified by length properties lower C_i / upper C_i (i=1, 2) on *EDMDCurveSet2D*.

Geometry with thickness is defined to have round ends and rounded corners such that it can represent a drill path.

Figure 33 demonstrates the usage of the parameters Upper/Lower of *CurveSet2D* and non-zero thickness. The left figure describes a solid created by a polyline C^*_1 from point p1 to p2 and thickness=D !=0. The right solid is described by a B-Spine C^*_2 and thickness bigger than 0.

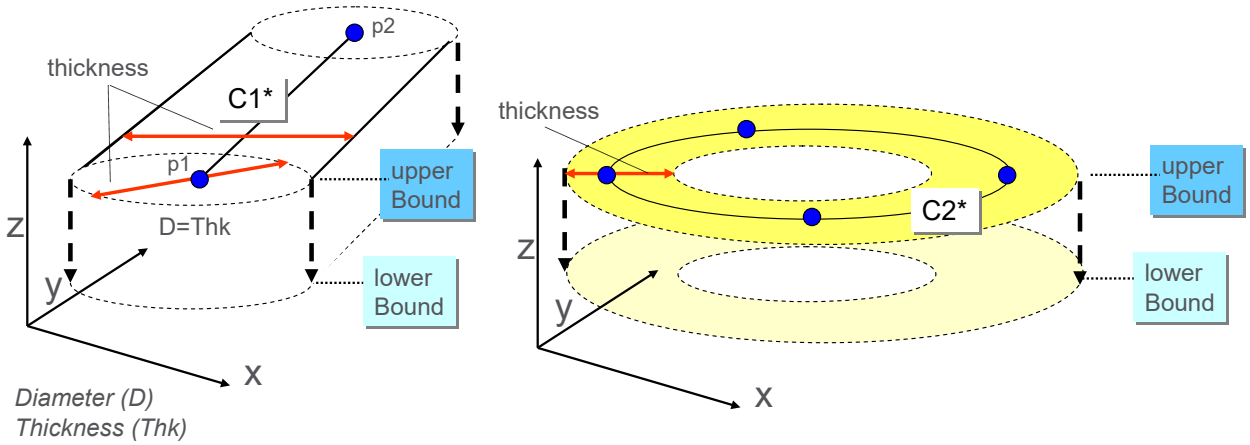


Figure 33: Usage of upper/lower bound and thickness

The bodies are described regarding their coordinate system. The element *EDMDTransformation* allows transforming the bodies in relation to a common coordinate system. Thus, this element describes the transformation between *EDMDShapeElement* coordinate system and the *EDMDCurveSet2D* coordinate system. If this element is not set, an identity matrix is assumed (so no transformation is needed). In this example two transformations (T1, T2) are needed.

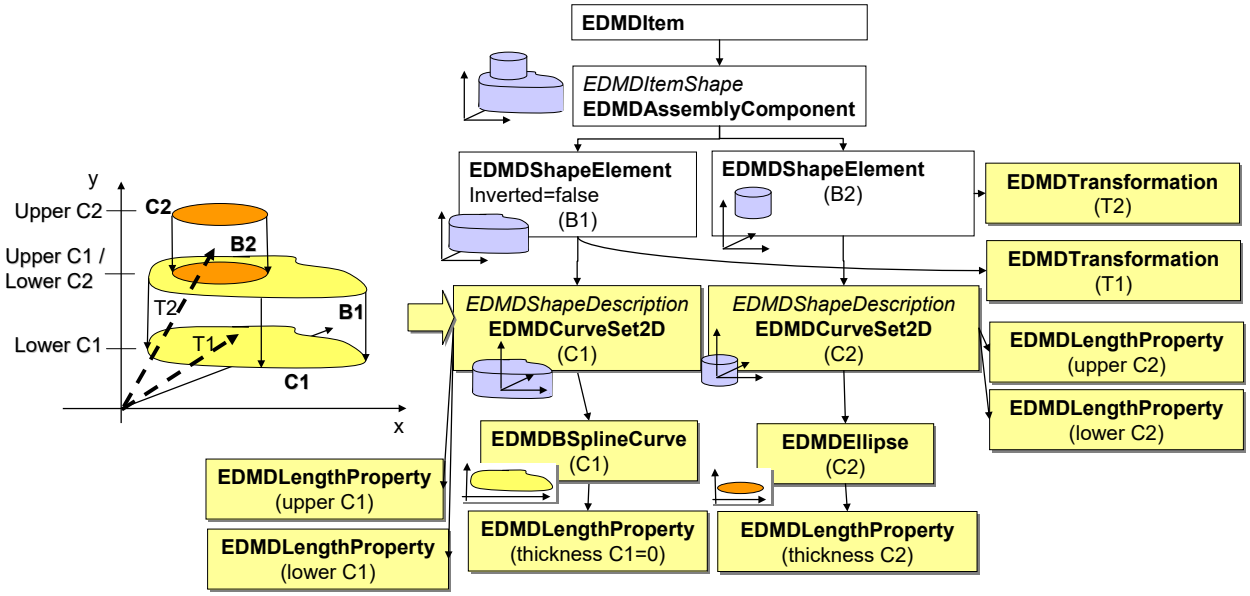


Figure 34: Instantiation of combination of "translation bodies" (example)

The specification of shapes in IDX is according to Constructive Solid Geometry (CSG). Constructive solid geometry allows a modeller to create a complex surface or object by using Boolean operators to combine objects. The attribute 'Inverted' for *ShapeElement* indicates whether the element is added to the shape of the item (OR operator), in which case inverted is 'false' or empty, or whether the element is taken away from the other objects (NAND operator), in which case inverted is 'true'. Figure 35 visualizes the NAND operator for specifying the shape of an item. The shape is put together in the order in which the elements occur in the XML document. The instantiation is same as in Figure 34 except the attribute value "Inverted" in *ShapeElement* B2.

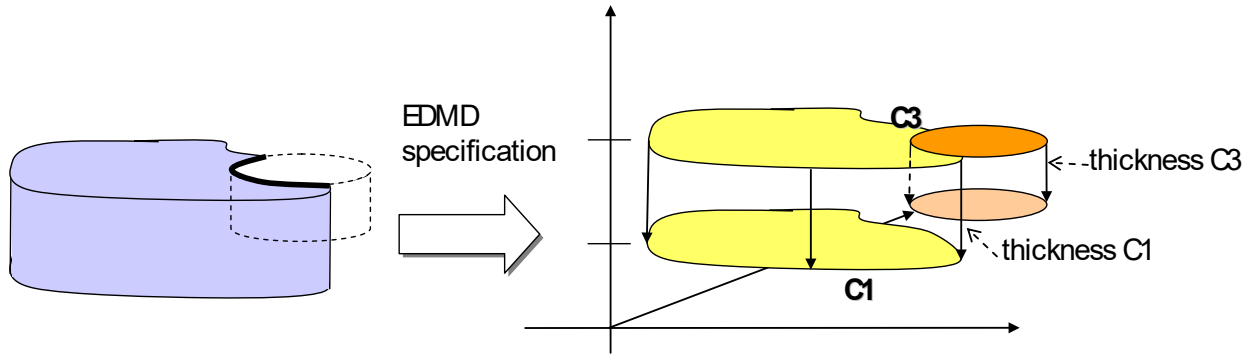


Figure 35: Representation of a shape of an item by "cut-out" another body ("NAND" operator)

There are several ways of describing a 2D curve in EDMD schema, such as B-Spline, arc curve etc. Depending on the topology of the shape to be approximated, a certain type is more capable than others. Thus, the IDX schema allows describing a curve as a composition of several curves (with different types). Depending on the shape it may also make sense to combine several curves of the same type (E.g. combine two arc curves with different radius). In the example shown in Figure 36, the main curve c1 is a composition of three curves of different types: B-Spline (c1-1), poly line (c1-2) and arc curve (c1-3).

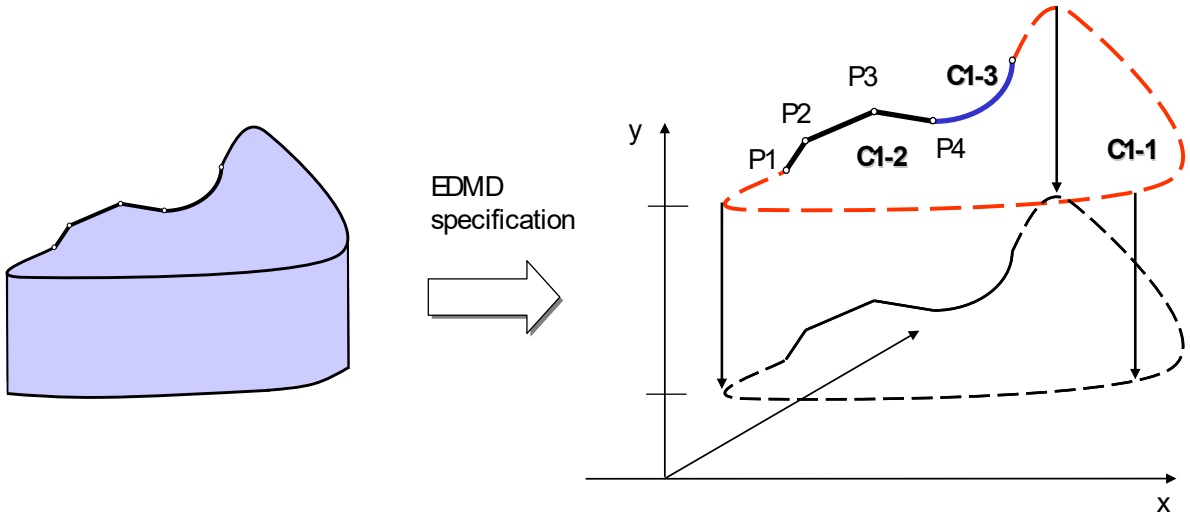


Figure 36: Describe a closed curve of by combination of several other curves

An exemplary instantiation of a shape described by a composite curve is shown in Figure 37. The difference between the instantiation shown before is using the element *EDMDCompositeCurve*. This element combines the three curves C1-1, C1-2 and C1-3.

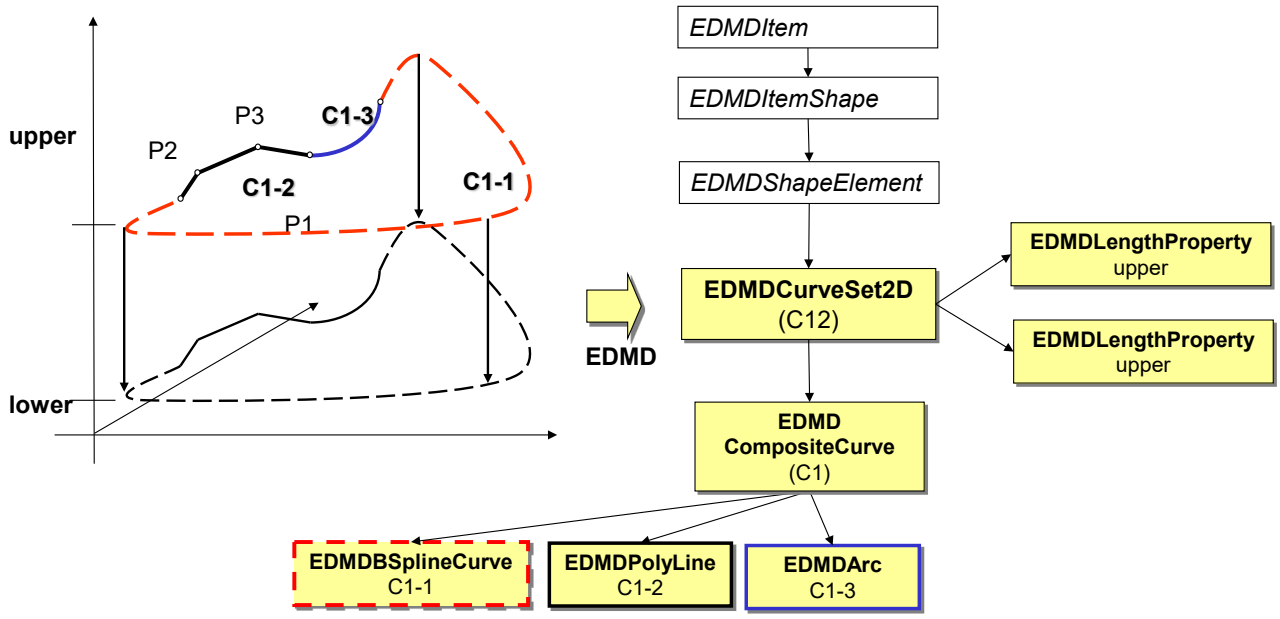


Figure 37: Body with a composite curve as footprint

Table 10 gives an overview on all curve types available in IDX schema.

EDMD schema name	Description	Comment
EDMDCircle3Point	An *EDMDCircle3Point* is an *EDMDCircle*, which is defined by three points.	
EDMDCircleCenter	An *EDMDCircleCircle* is an *EDMDCircle*, which is defined by a center and a diameter.	
EDMDBSplineCurve	An is an *EDMDCurve* and is a piecewise parametric polynomial or rational curve described in terms of control points and basis functions. The B-spline curve has been selected as the most stable format to represent all types of polynomial or rational parametric curves. With appropriate attribute values it is capable of representing single span or spline curves of explicit polynomial, rational, Bézier or B-spline type. For more details take a look to ISO 10303-42.	**Defines additional attributes for information:** **CurveForm[1] (Used to identify particular types of curve)** **ClosedCurve (Indication of whether the curve is closed),** **SelfIntersect (Flag to indicate whether the curve self-intersects or not)**
EDMDEllipse	An Ellipse is a conic section defined by the lengths of the semi-major and semi-minor diameters and by the position (centre or midpoint of the line joining the foci) and orientation of the curve.	
EDMDParabola	A parabola is an open conic curve defined by its focus and apex.	
EDMDPolyLine	A Polyline is a curve that is represented by n-1 linear segments, defined by a list of n points. A Polyline may be closed and may self-intersect.	
EDMDOffsetCurve	An *EDMDOffsetCurve* is a curve at a constant distance from a basis curve in two-dimensional space. This entity defines a simple plane-offset curve by offsetting by distance along the normal to BasisCurve in the plane of basis_curve. For more details take a look to ISO 10303-42.	***Not part of* IDX *conformance class 1 and implementer guide.***
EDMDTrimmedCurve	An *EDMDTrimmedCurve* is a curve of finite length that is the result of trimming a curve between two points. The trimming points are the start and end points of the *EDMDTrimmedCurve*. These points are specified by *EDMDPoints*.	***Not part of* IDX *conformance class 1 and implementer guide.***
EDMDCompositeCurve	**An *EDMDCompositeCurve* is a curve that is composed of a number of curves which join end to end.**	**Despite other elements in this table this curve does refer to other curves.**

Table 10: Overview on available curve types

[1] Following curve forms may be identified: *PolylineForm, CircularArc, EllipticArc, ParabolicArc, HyperbolicArc, Unspecified*

7.1.1 XML representation of an *EDMDArc*

Figure 38 shows the XSD schema and exemplary XML representation of an *EDMDArc*. In case the include angle is negative, the arc describes a clockwise path from start point to end point (otherwise anticlockwise).

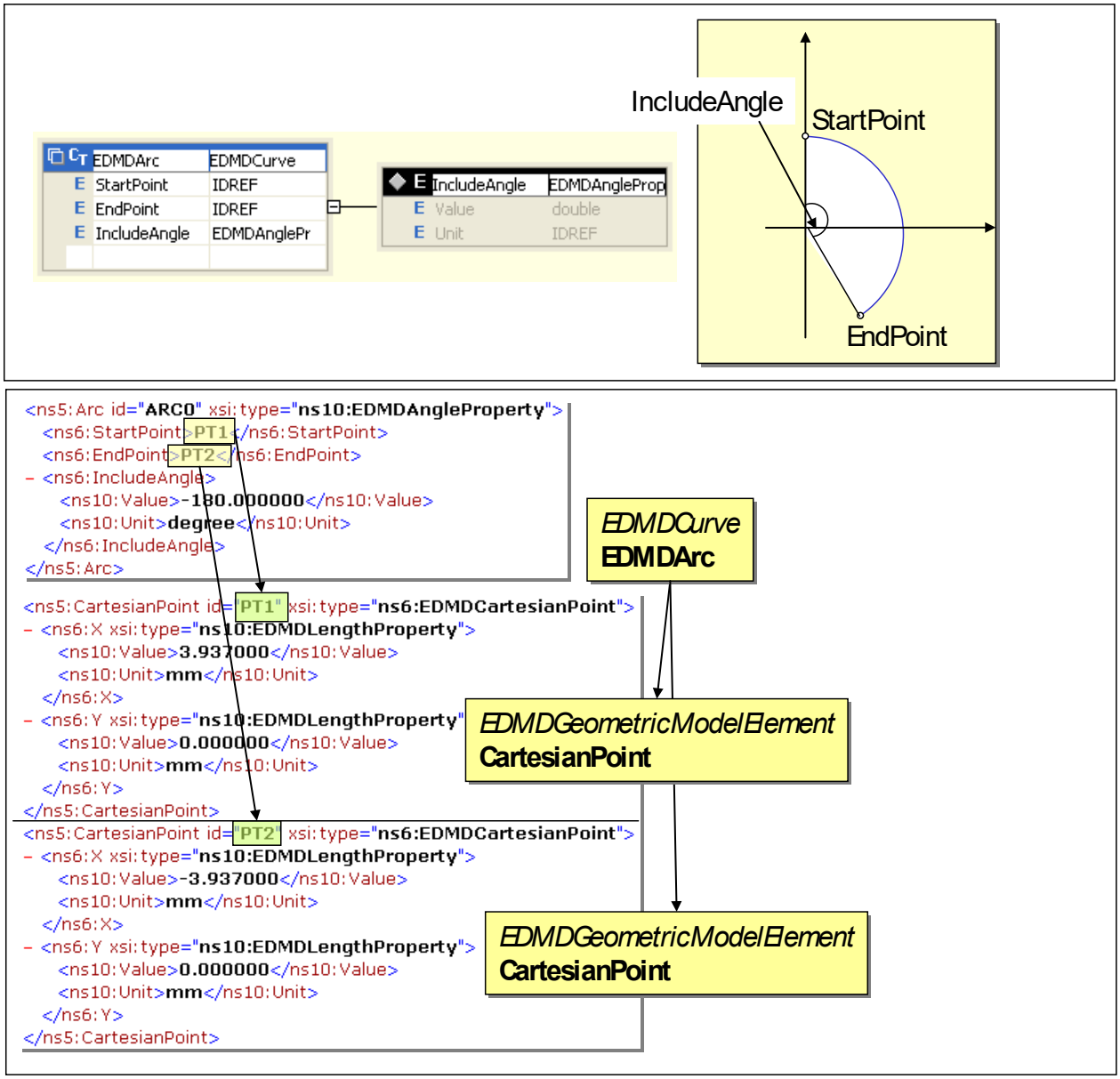


Figure 38: XML representation of an arc

7.1.2 XML representation of an *EDMDBSplineCurve*

Figure 39 shows the XSD schema XML representation of an *EDMDBSplineCurve*.

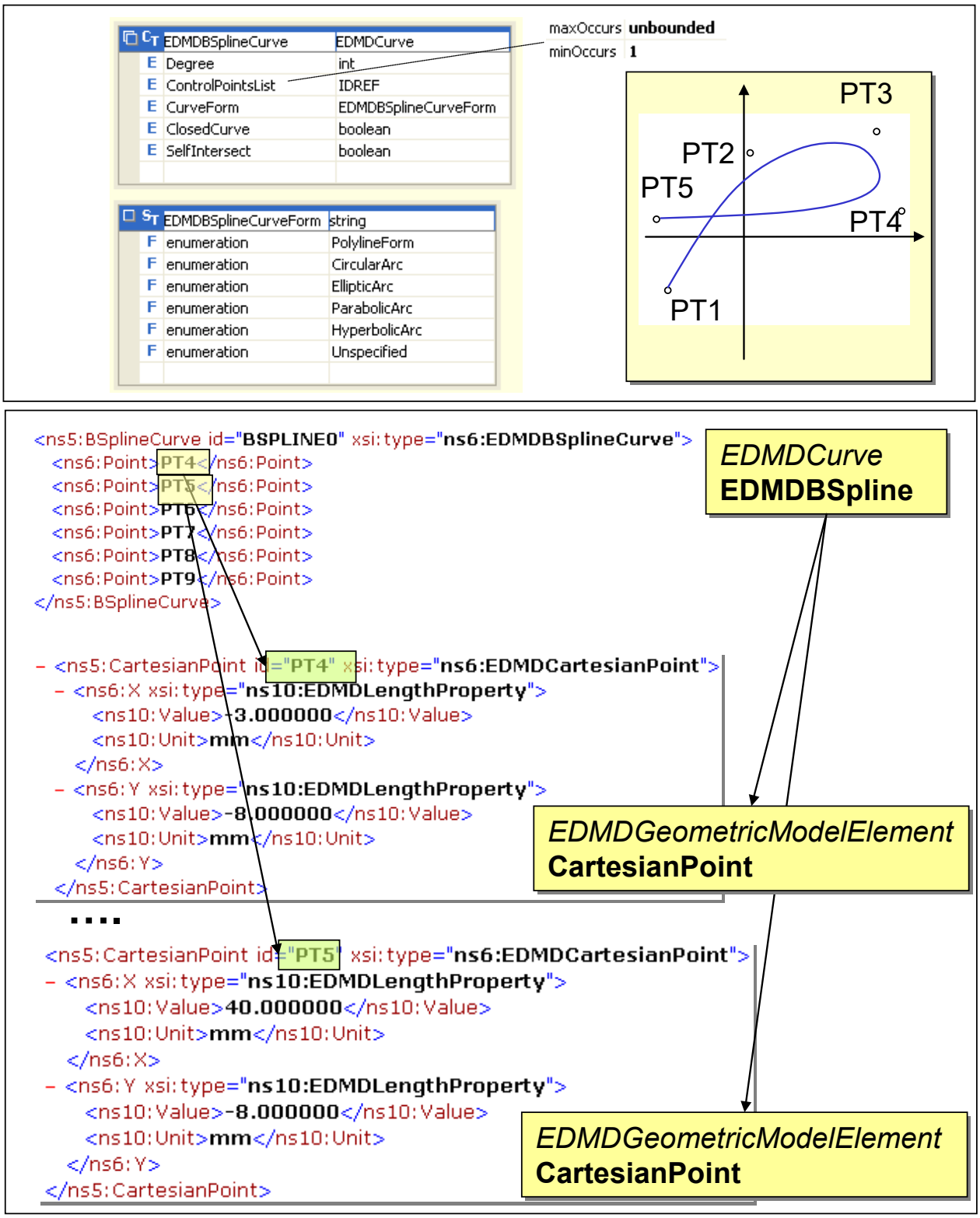


Figure 39: XML representation of a B-Spline

7.1.3 Instantiation of an *EDMDCircle3Point*

Figure 40 shows the XSD schema XML representation of an *EDMDCircle3Point*. In case three points are not placed in one line, they describe a circle.

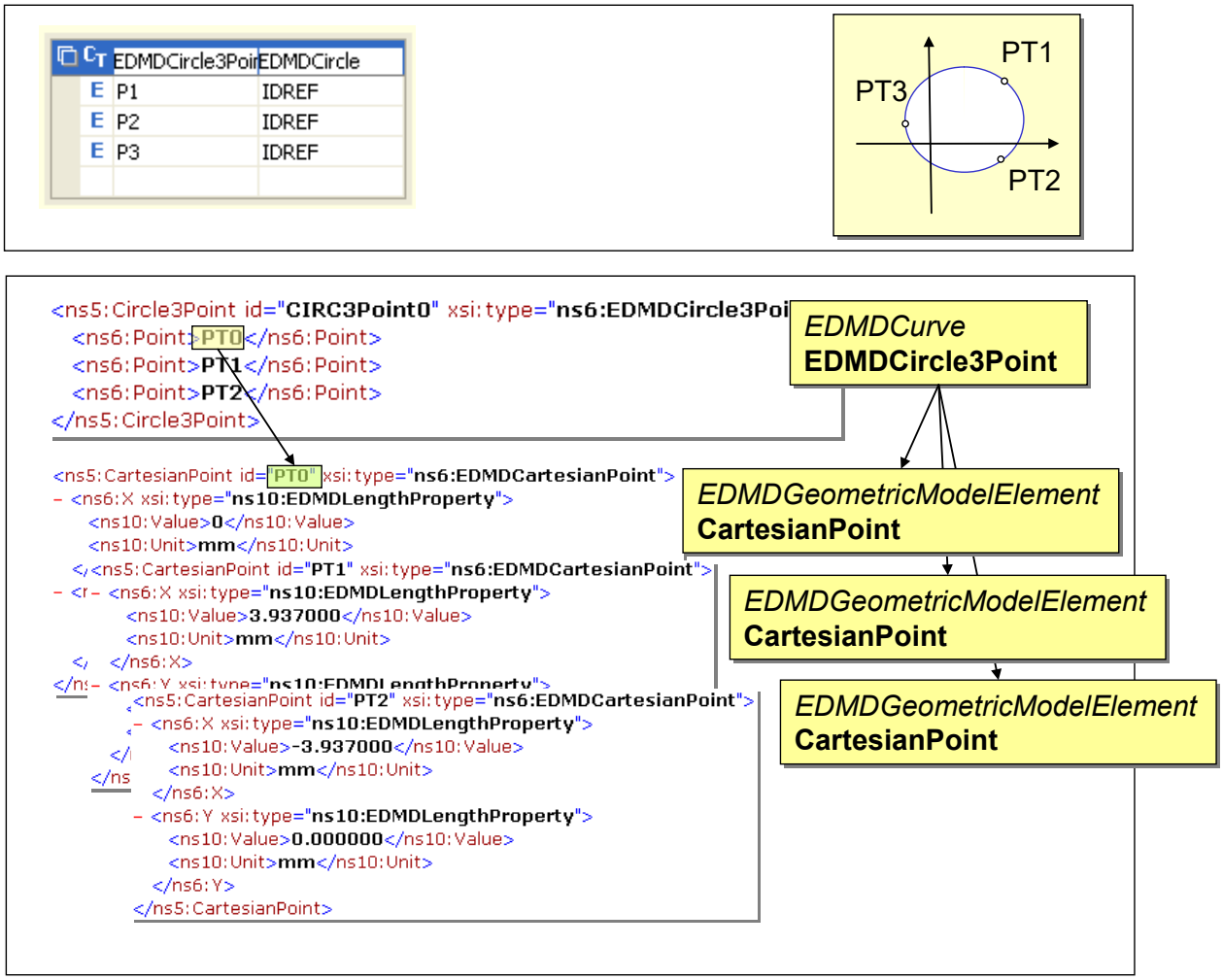


Figure 40: XML representation of a circle described by three points

7.1.4 XML representation of an *EDMDCircleCenter*

Figure 41 shows the XSD schema XML representation of an *EDMDCircleCenter*. Here the circle is described by its center point and diameter.

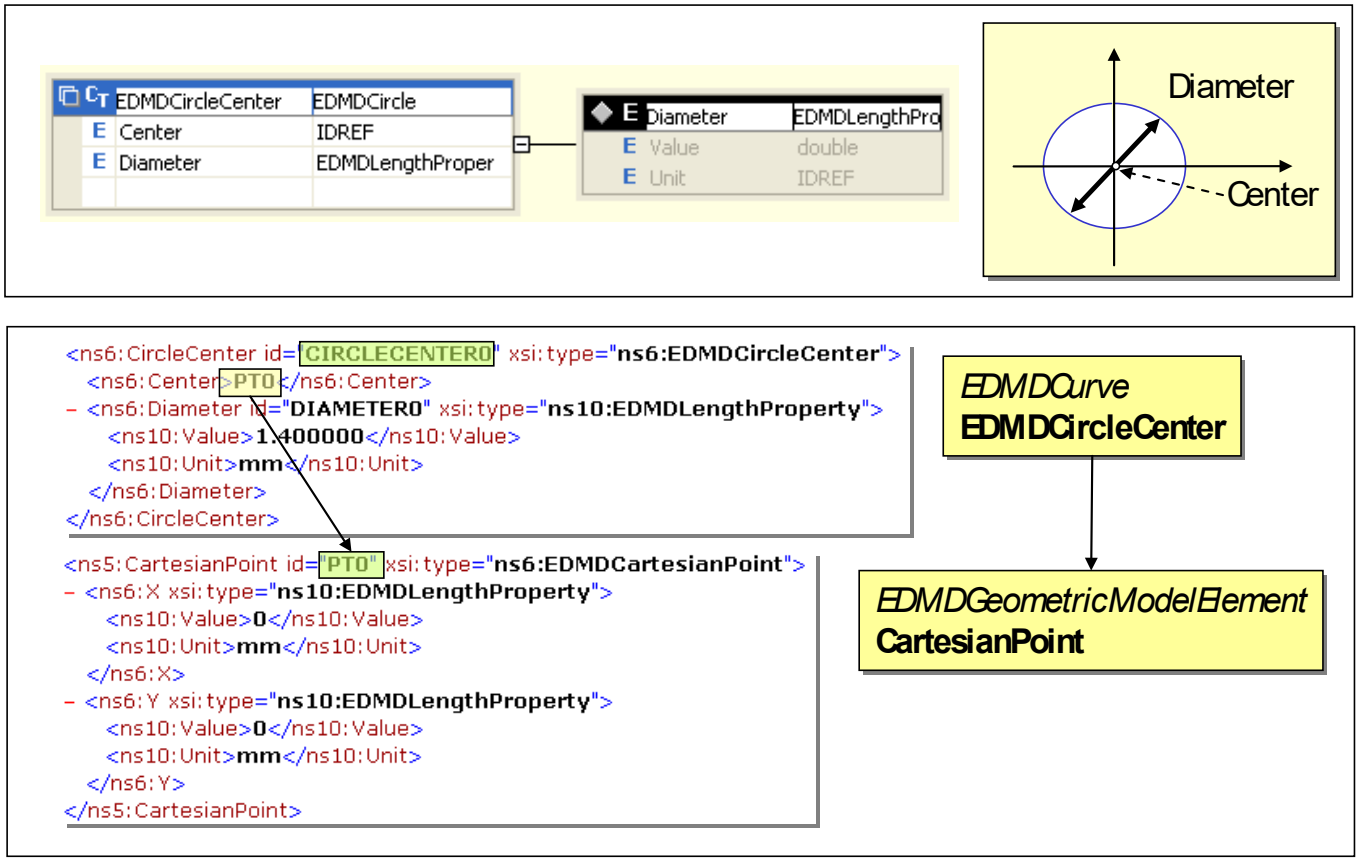


Figure 41: XML representation of a circle described by center point and diameter

7.1.5 XML representation of an *EDMDEllipse*

Figure 42 shows the XSD schema XML representation of an *EDMDEllipse*. The ellipse is described by its center point and the length of its semi major and semi minor.

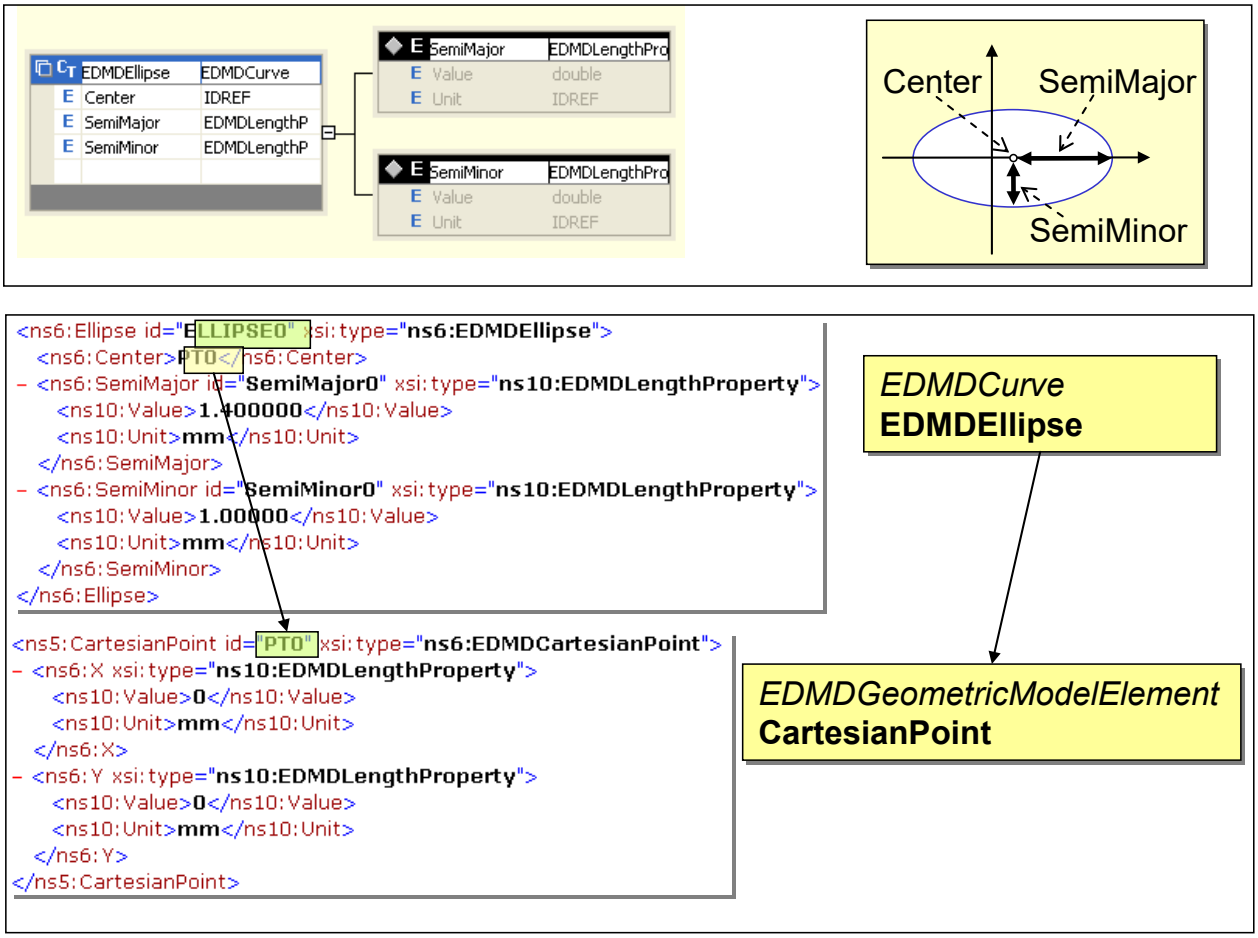


Figure 42: XML representation of an Ellipse

7.1.6 XML representation of an *EDMDParabola*

Figure 43 shows the XSD schema XML representation of an *EDMDParabola*. The parabola is described by its focus and apex.

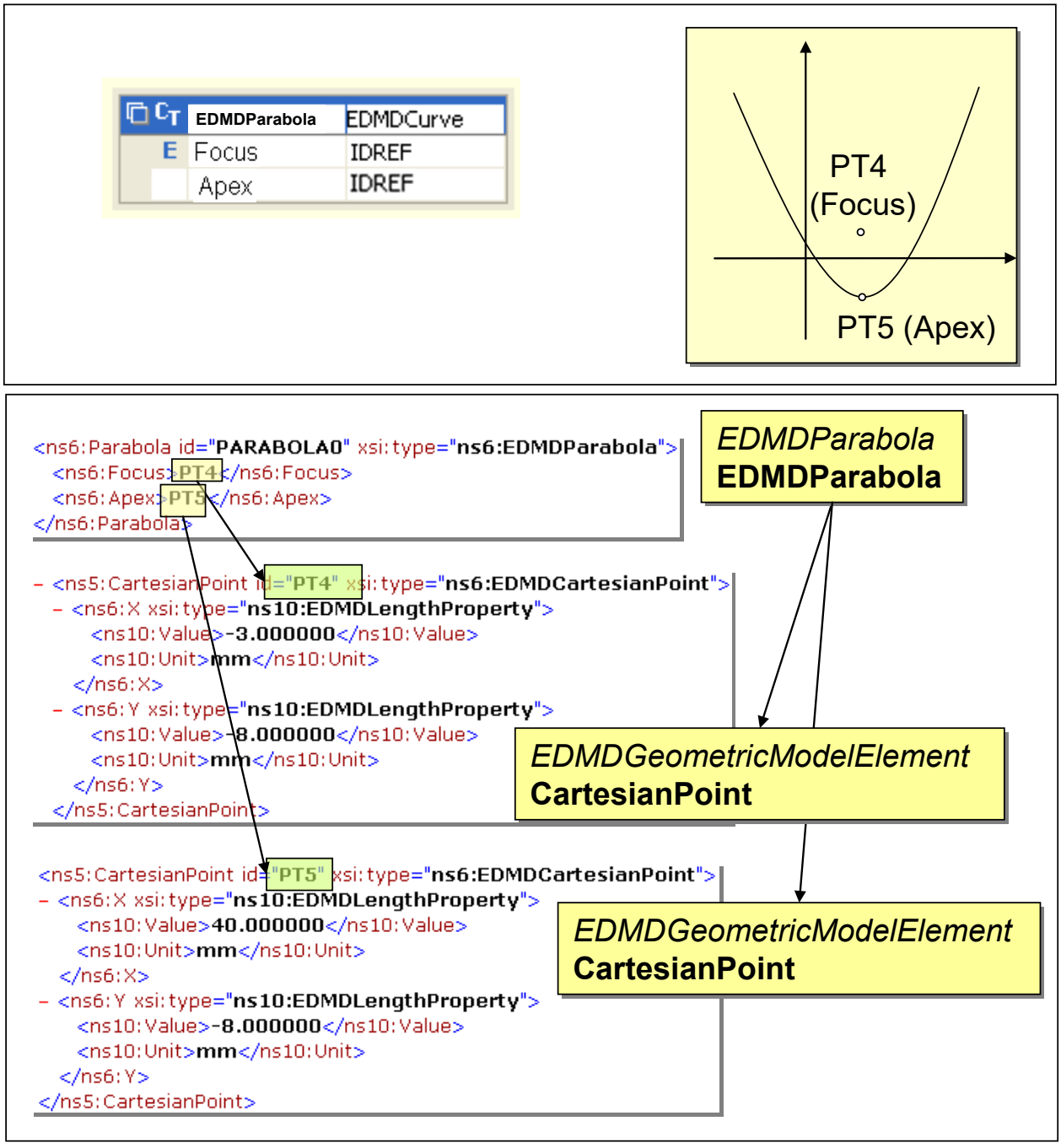


Figure 43: XML representation of a parabola

EDMDParabola only makes sense when referenced by an *EDMDTrimmedCurve* entity.

7.1.7 XML representation of an *EDMDPolyLine*

Figure 44 shows the XML representation of an *EDMDPolyLine*. The polyline is described by a set of points.

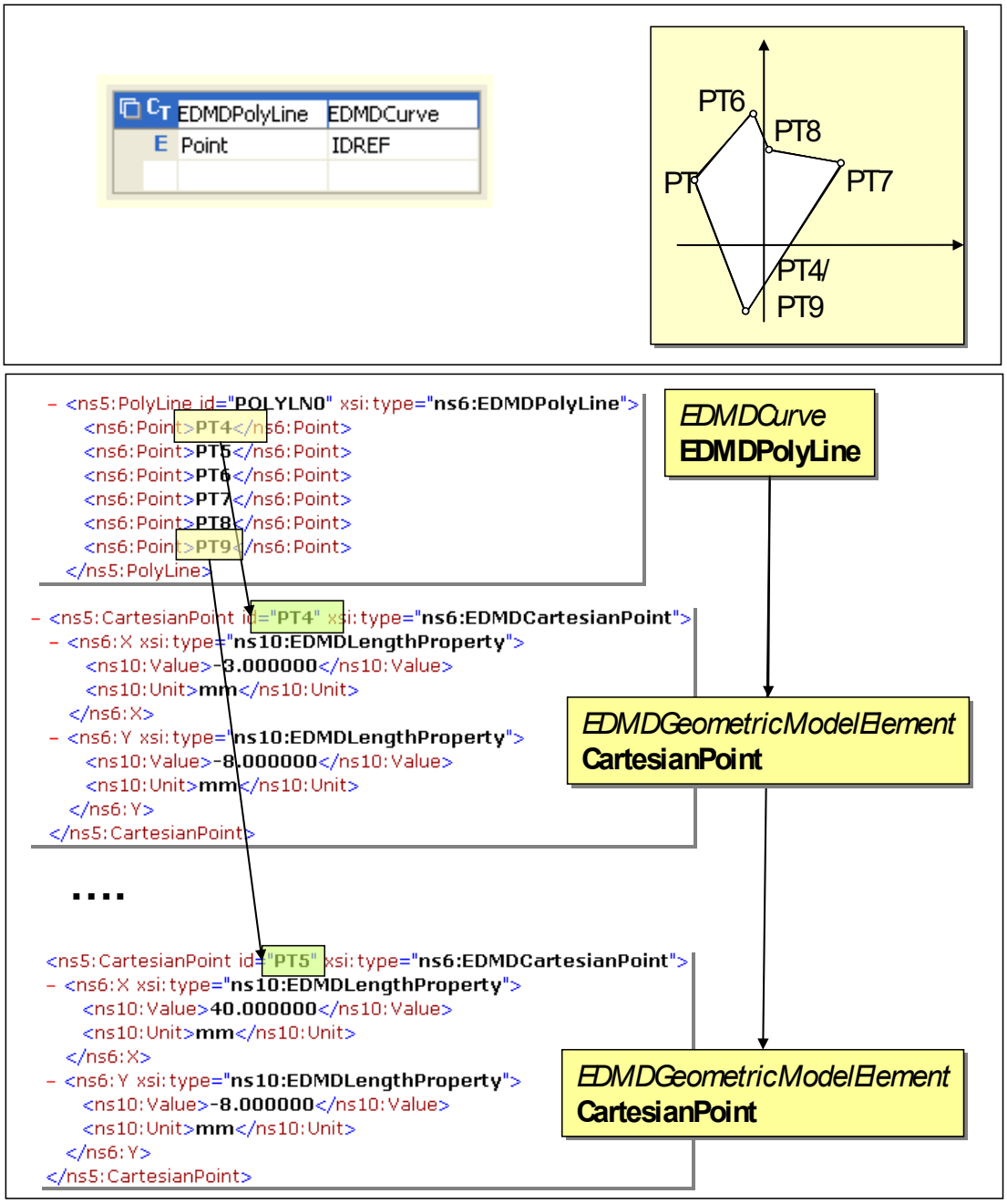


Figure 44: XML representation of a polyline

7.1.8 XML representation of an *EDMDCompositeCurve*

A composite curve consists of several sections. Each section is described by a curve which is supported by the IDX schema. Figure 45 shows a composite curve which consist of a poly line, arc and B-Spline.

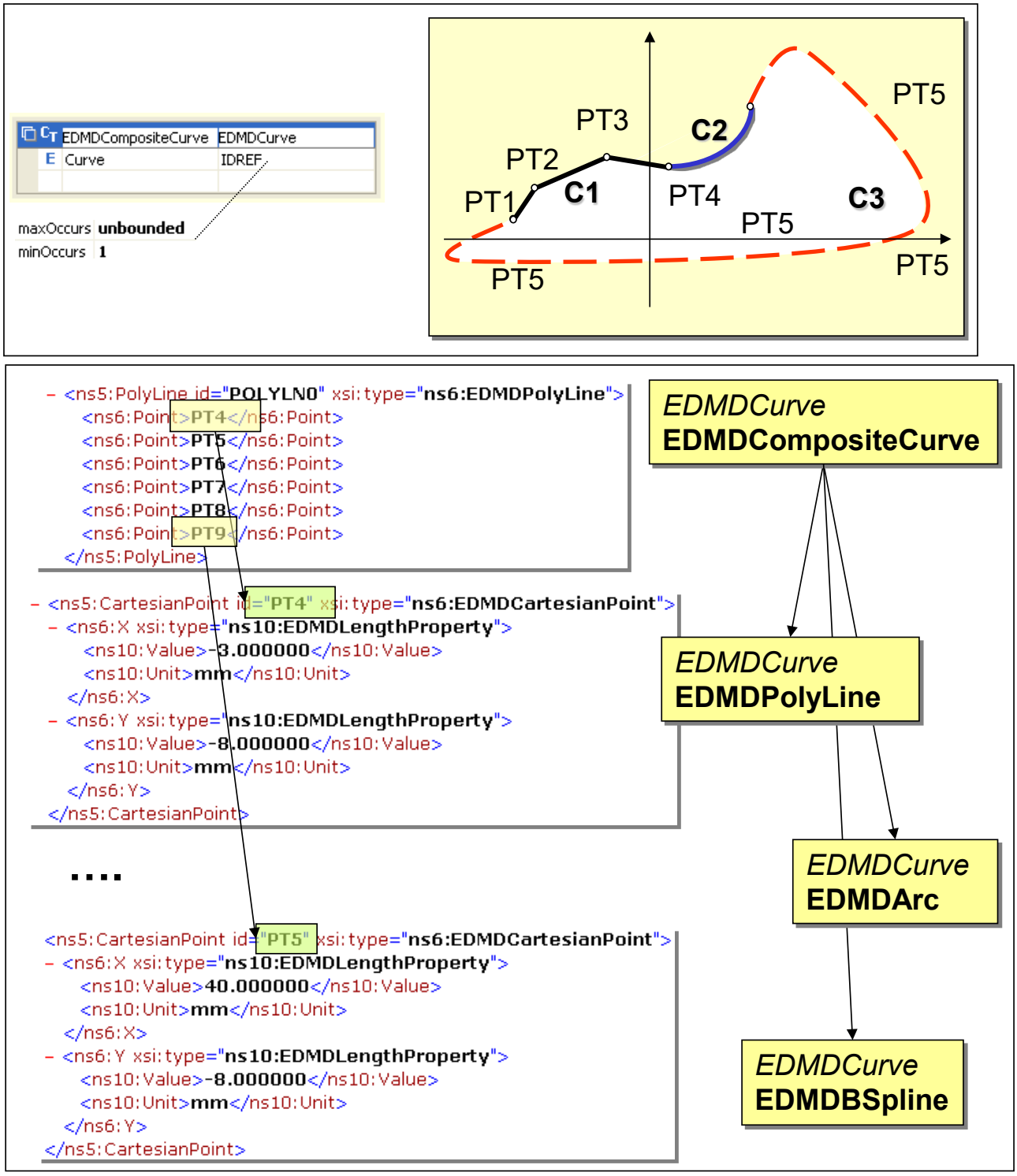


Figure 45: XML representation of a composite curve

7.1.9 Representation of the shape of an item as external file

Figure 46 shows the XSD schema for representing the shape of an item as external file. The attribute Location describes the full qualified URI of the location to access the external file. E.g. file://~/example/sample.file http://www.domain.tld/service. If the access mechanism needs additional parameters, *EDMDSource* could be the source of multiple files. Local *EDMDUserProperties* and the *EDMDUserProperties* of the referencing *EDMDExtensibleObject* are passed.

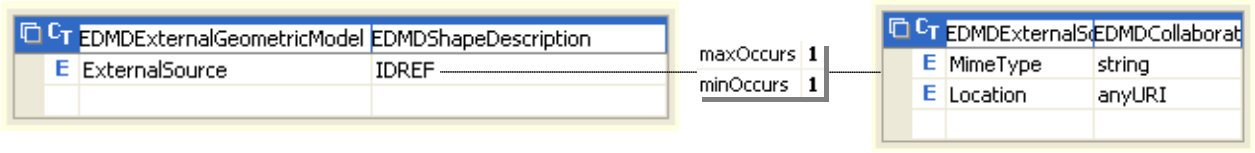


Figure 46: XSD schema for representing the shape of an item as external file

Figure 47 shows the instantiation of an item with external shape representation.

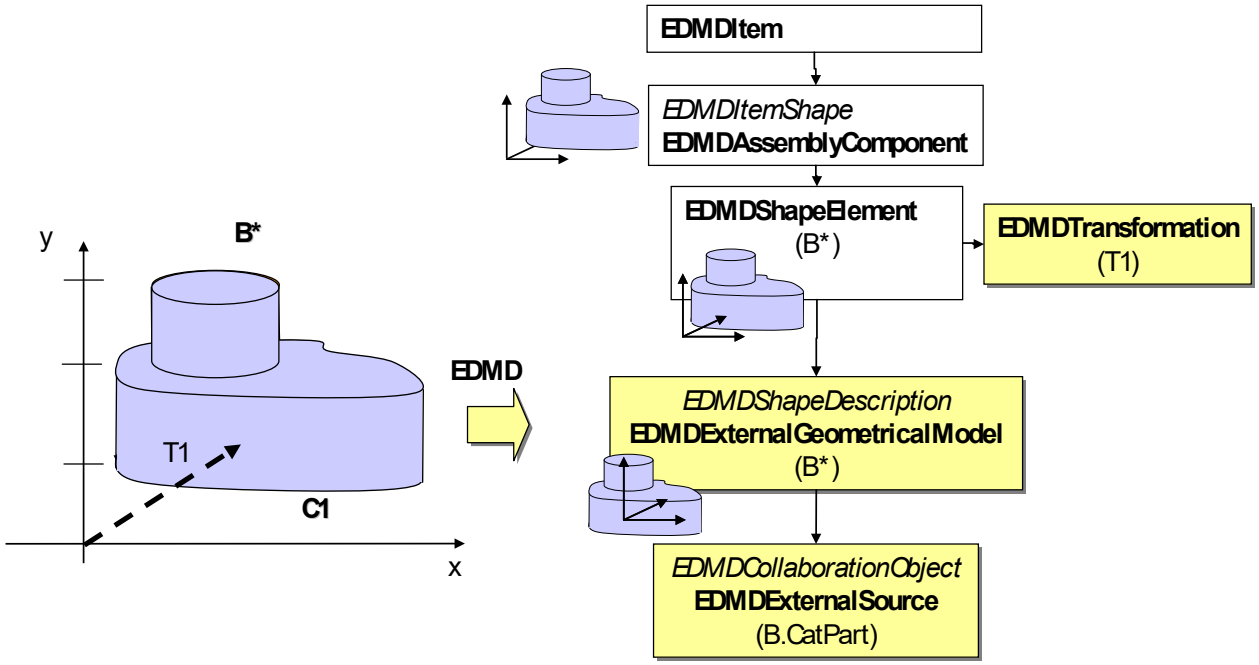


Figure 47: Instantiation for implicit representation of an item's shape

7.1.10 XML representation of items of a PCB board (example)

This section describes the representation of a PCB board (see Figure 29) with components and design features in XML. This is described based on an exemplary instantiation of an item of a board in section 0. How to assemble these elements to a PCB board will be shown in section 0.

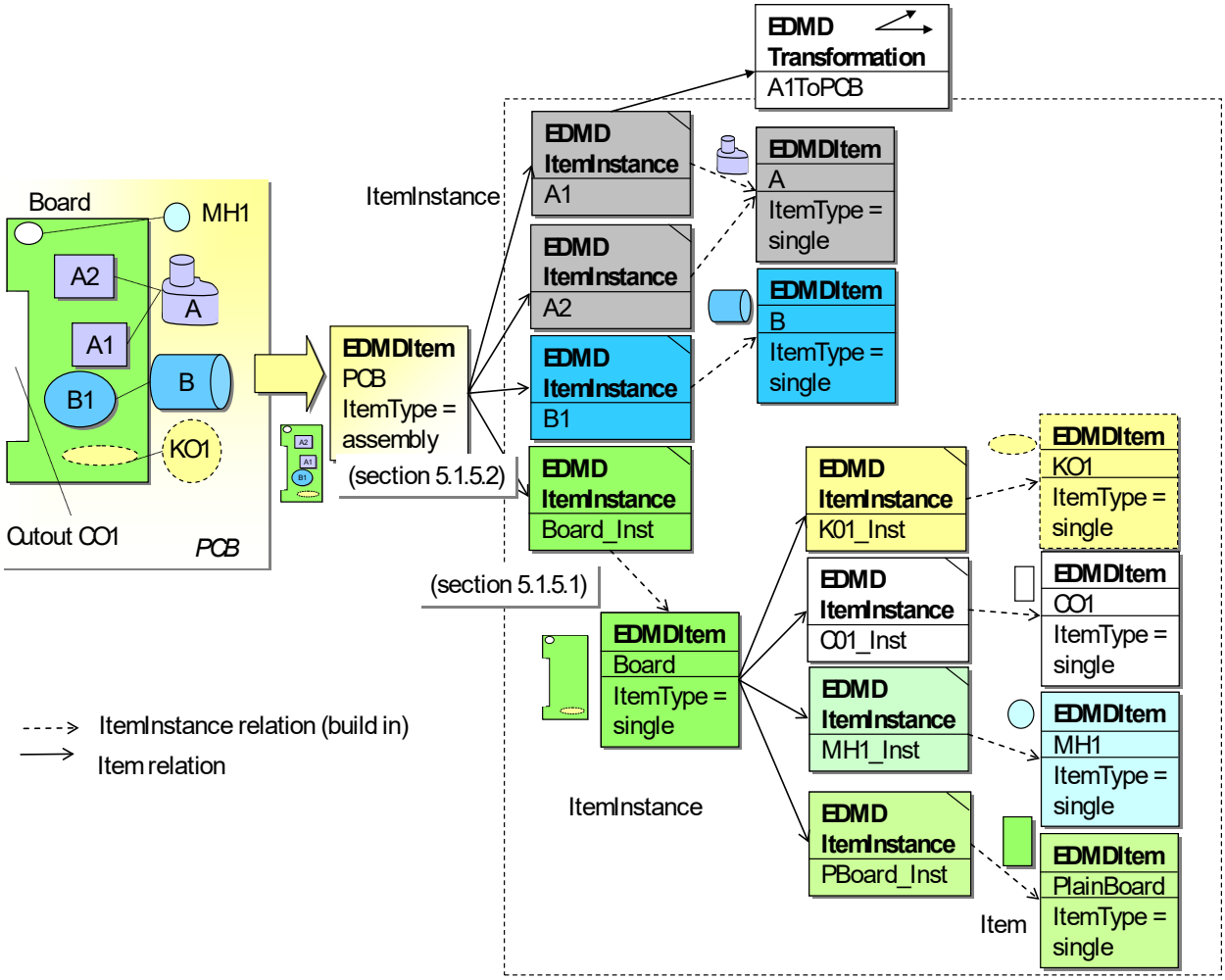


Figure 48: Overview: Instantiation of the board

7.1.10.1 Exemplary XML representation of an item

This section describes the instantiation of an item of a board with all elements needed. This is shown based on a mounting hole (see Figure 49). The concrete subclass of the item shape is *InterFeatureStratum*. The attribute *InterFeatureStratumType* is set to "*PartMountingFeature*" and the value of *ShapeElementType* is "Cutout". Since the mounting hole is on the board and therefore has no height, the lower bound and upper bound value of the *EDMDCurveSet2D* is set to 0 mm. The shape is described by a circle with a diameter (*EDMDCircleCenter*) of 1.4 mm. In case another shape description is needed, section 0 contains all supported shape representations.

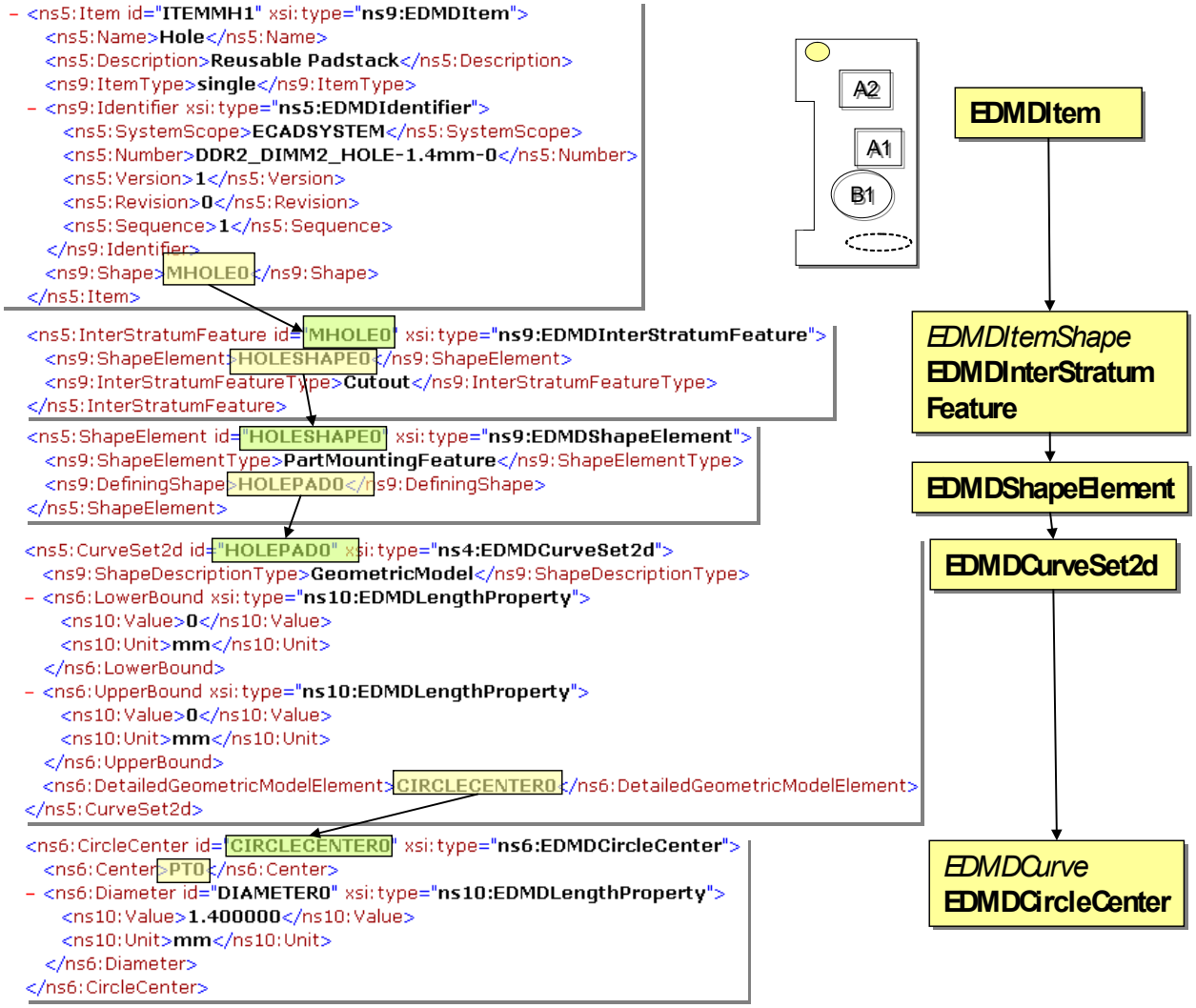


Figure 49: XML representation of a mounting hole

The other items are represented accordingly. The IDX types and attributes needs to be selected based on Table 3.

7.1.10.2 XML representation of PCB board with components (assembly)

The entire PCB board consists of PCB components and the board assembly. The board assembly groups the plain board and its features (E.g. cut-out, mounting hole) and restrictions (keep-in, keep-out). The position of the components is described relatively to the coordinate system of the PCB board assembly by IDX

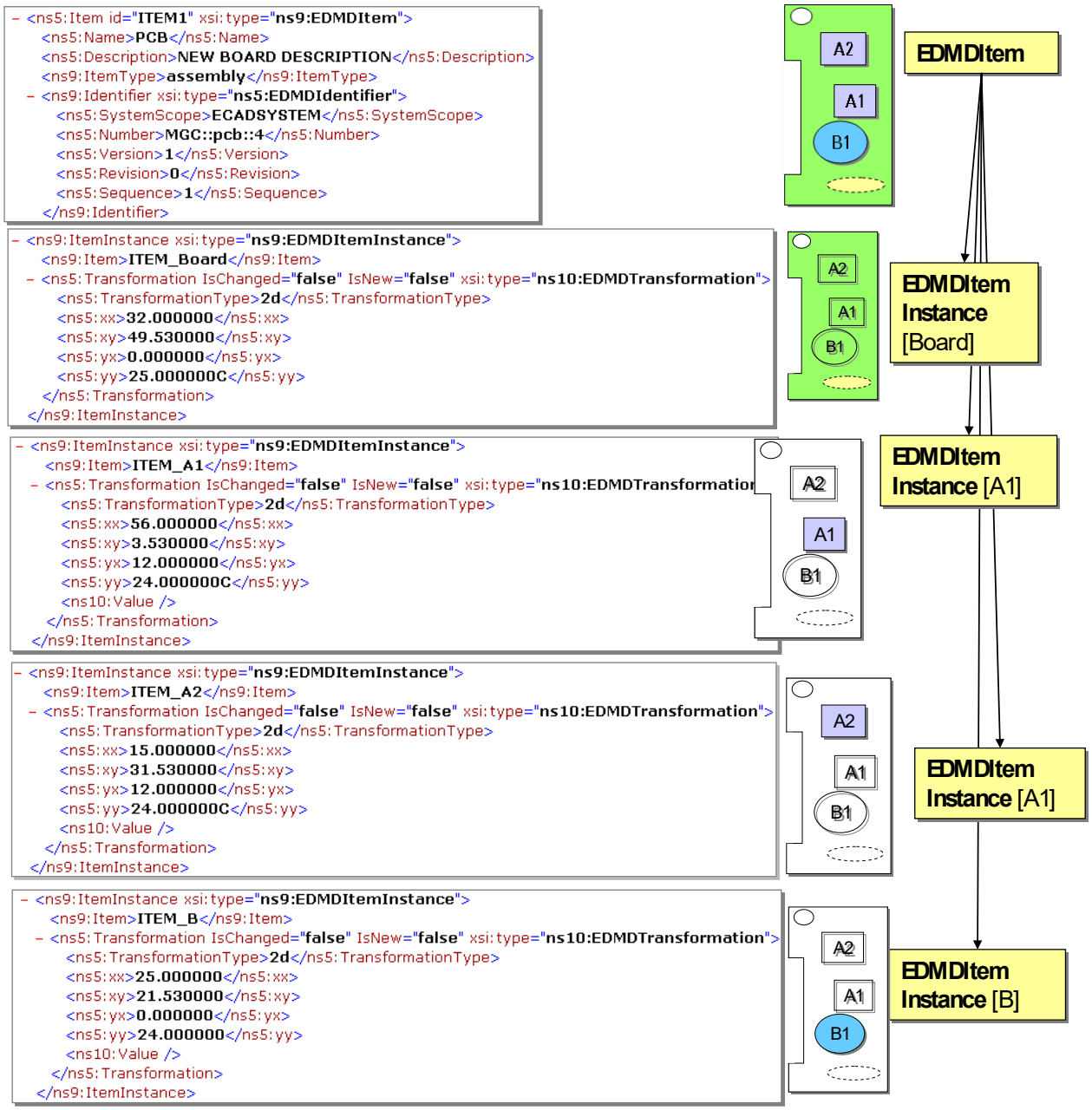


Figure 50: Representation of a board with components

8 Instantiation of test cases

8.1 Test Case 1-1: “Support of shape representations”

IDX allows the explicit specification of a shape of an item based on translation bodies. A translation body is described by its footprint (2D curve) and its height (thickness). There are several kinds of curves specified in IDX schema. Each type should occur once in the baseline specification. The shape may be constructed using Boolean operations (Constructive Solid Geometry, CSG). A shape which has all operations should occur at least once in the baseline.

In order to check also for non-PCB elements, at least two keep-out areas should be on the board and described “explicit”. This test case may be realized by defining a baseline as shown in Figure 51. That means define 9 items on a PCB board.

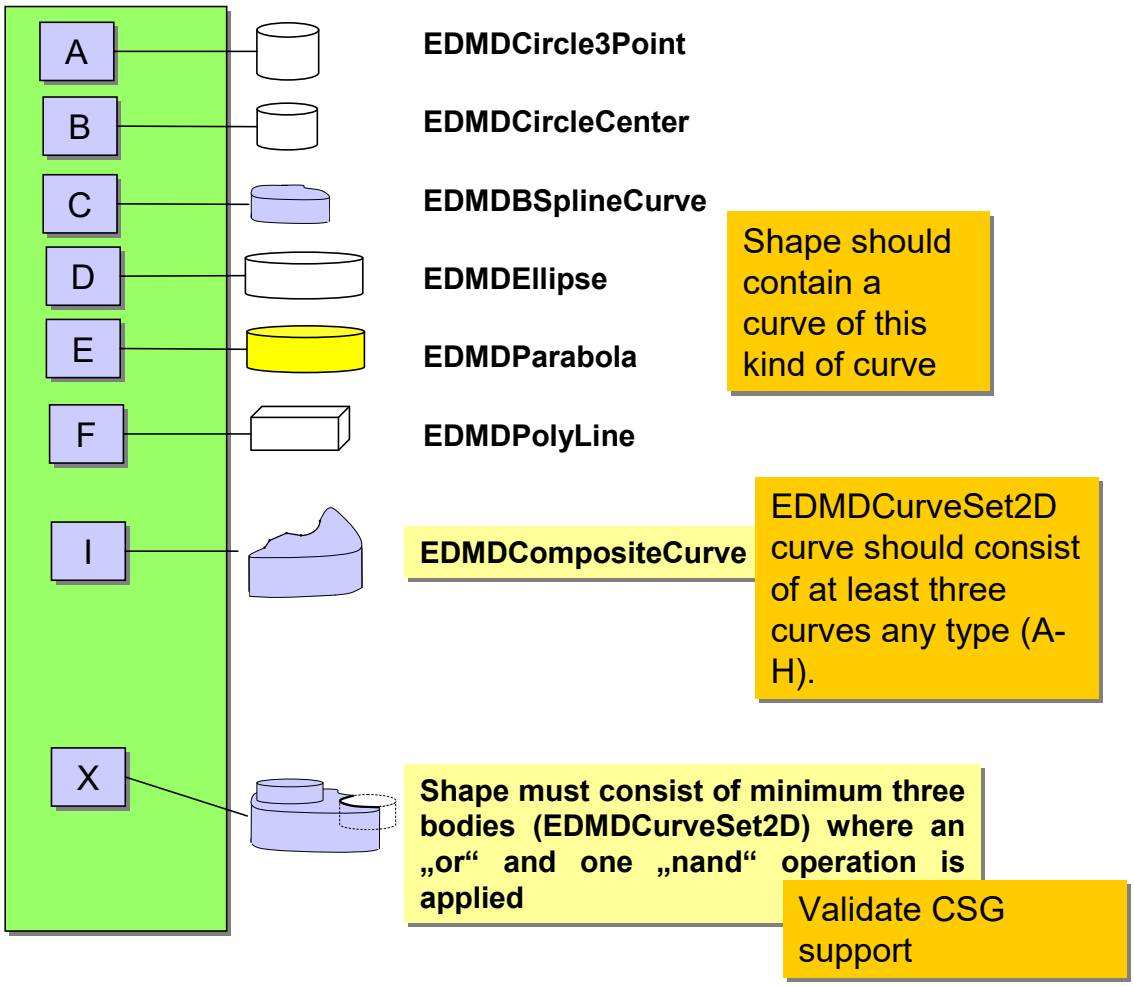


Figure 51: Verify support of different kinds of “explicit” shape representation

A possible XML representation of this test case could be derived based on section 0 (different types of geometry description) and section 7.1.9 (description of an item instance with its geometrical representation).

8.2 Test Case 1-2: “Definition of board baseline B”

Test case 1-2 covers the definition of a board baseline which contains PCB components (E.g. capacitor, microcontroller) and non-PCB components (E.g. keep-out and mounting hole) as well as changes executed on this baseline. A possible XML representation of this baseline is outlined in section 7.1.10.

This should cover following cases:

- There are several instances of one PCB component.

- The shape of at least one PCB component is represented implicit (by external file) and one explicit (by 2.5 D geometry).
- Mounting hole is defined by circle curve and ellipse curve.
- At least one keep-out area should be represented implicit and one explicit.

This covers that all possible collaboration items may be part of a baseline and both ways of representing a shape are supported.

Figure 52 shows a possible constellation of test case 1-2. The shape of PCB component A is represented explicit; the shape of PCB component is described by an external file. Additionally, the baseline defines a mounting hole and keep-out-area.

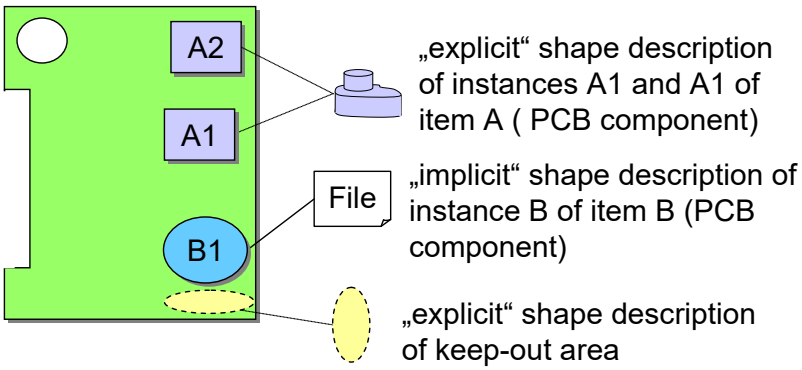


Figure 52: Verify definition of initial base line which contains main item types for collaboration

This baseline is used as starting point for all changes described in following sections.

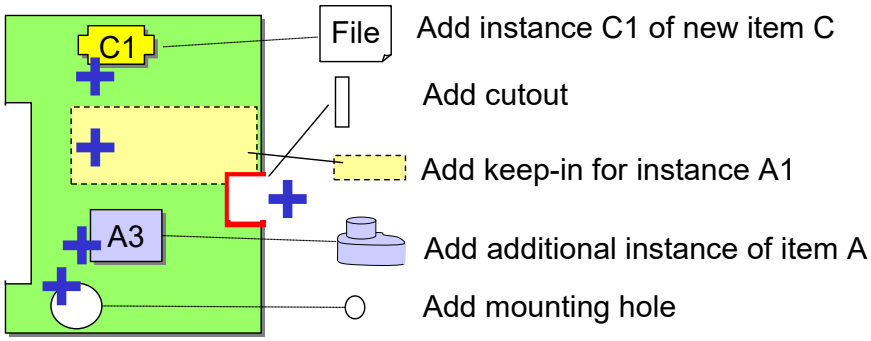
8.3 Test Case 2-1: Addition of new items (Change 1)

Test Case 2-1 contains following changes:

- Add instances of PCB components (E.G. capacitor) which exist in collaboration space and one which does not. This should cover both cases: shape of the component is represented implicit (by external file) and explicit (by 2.5 D geometry).
- Add new non-PCB components like cut-out, keep-in and mounting hole. This should cover both cases: shape of the component is represented implicit (by external file) or explicit (by 2.5 D geometry).

Figure 53 shows a possible constellation of test case 2-1:

Change 1:



Situation after change 1:

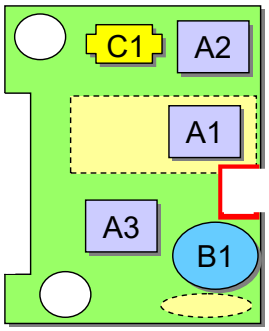


Figure 53: Verify addition of new items

8.3.1.1 How to represent addition of item instances?

The change section contains the new item of the board and link to previous version. Since the plain board without PCB components is described in a separate item (here: “board”), this item and its previous version is also listed in this section. The dataset section contains the modified item description with new item instances. The description of “PCB” contains an instance of the new “board” item and the PCB components C1 and A3. The “board” item contains the instances of the additional mounting hole, cut-out, and keep-in area.

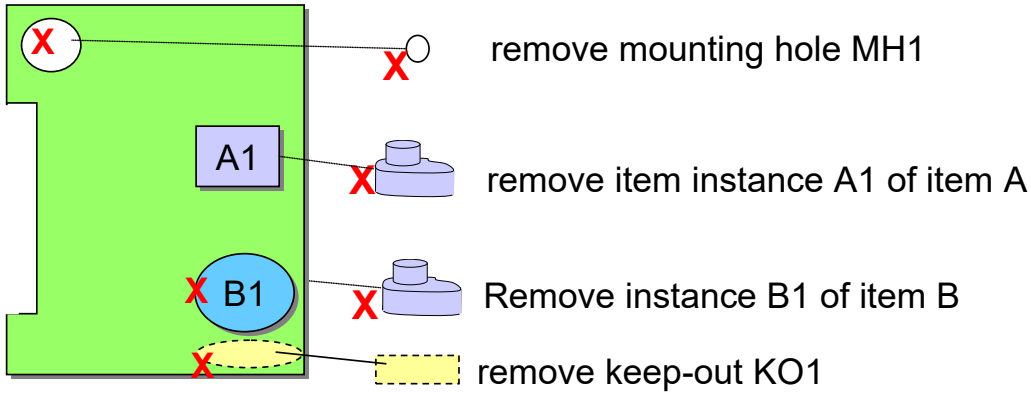
8.4 Test case 2-2: Removal of items (Change 2)

Remove instances of PCB components (E.G. capacitor). This should cover both cases: shape of the component is represented implicit (by external file) or explicit (by 2.5 D geometry).

Remove non-PCB components like cutout, keep-in and mounting hole.

Figure 54 shows a possible constellation of test case 2-2:

Change 2:



Situation after change 2:

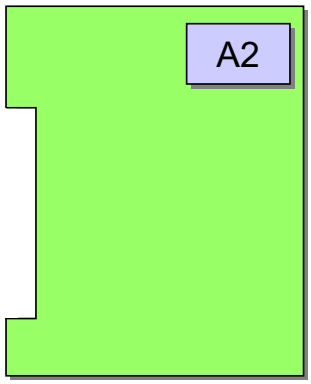


Figure 54: Verify removal of items from board

8.4.1.1 How to represent removal of item instances?

The change section contains the “PCB” and “board” item with the reference to previous version. Additionally, each deleted instance (MH1, A1, B1, KO1) is listed. The dataset section contains the new version of the assembly items PCB and board.

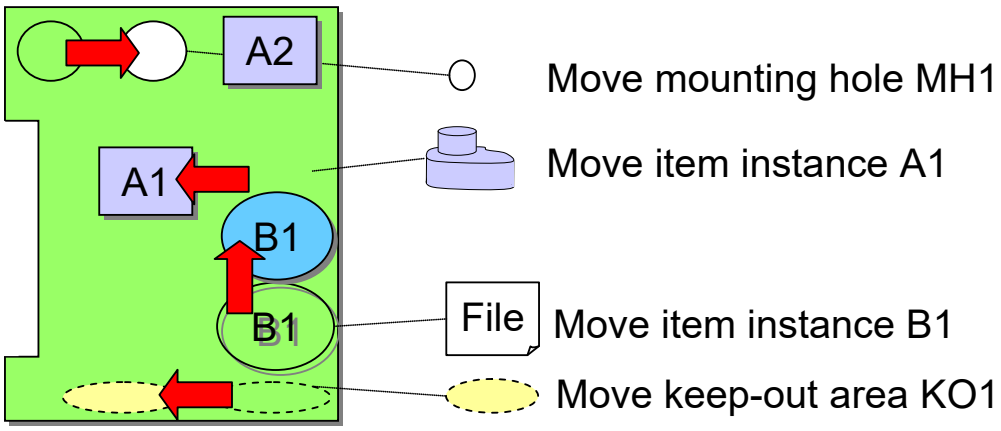
8.5 Test case 2-3: Movement of items (Change 3)

Change position of instances of PCB components (E.G. capacitor). This should cover both cases: shape of the component is represented implicit (by external file) or explicit (by 2.5 D geometry).

Change position of non-PCB components like cutout, keep-out and mounting hole.

Figure 55 shows a possible constellation of test case 2-3:

Change 3:



Situation after change 3:

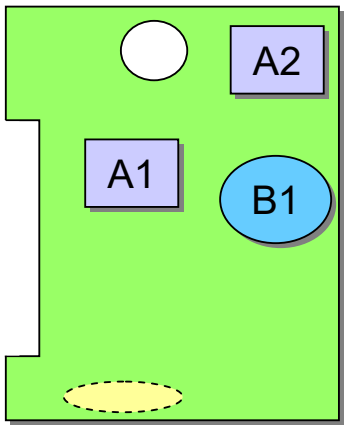


Figure 55: Verify movements of items

8.5.1.1 How to represent movement of item instances?

The change section contains the “PCB” and “board” item with the reference to previous version. The new position of the item instances relatively to its items (PCB or board) is described in the corresponding transformation section of the item instance: The item “PCB” contains the new position of A1, A2, B1 and the “board”. The “board” item contains the new position of keep-out KO1 and mounting hole MH1.

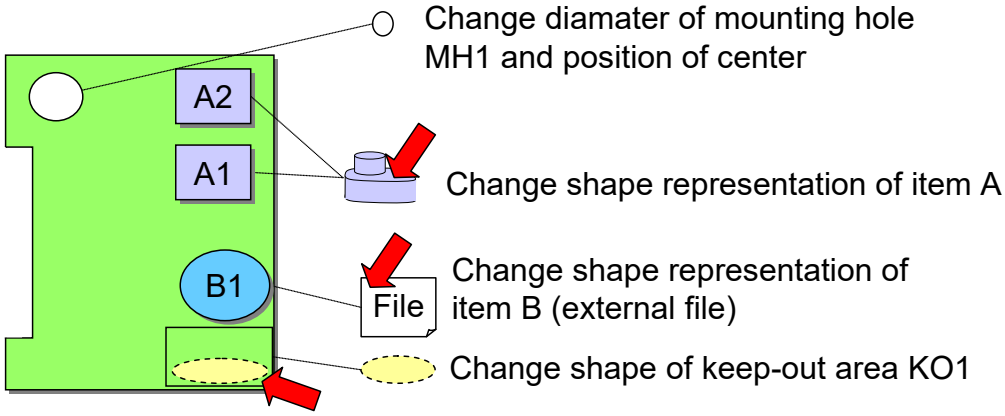
8.6 Test case 2-4: Modification of items (Change 4)

Change shape of instances of PCB components (E.G. capacitor). This should cover both cases: shape of the component is represented implicit (by external file) or explicit (by 2.5 D geometry).

Change shape of non-PCB components like cutout, keep-out and mounting hole.

Figure 56 shows a possible constellation of test case 2-4:

Change 4:



Situation after change 4:

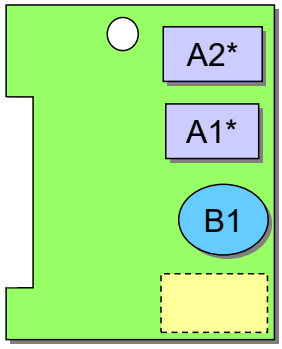


Figure 56: Verify modification of items

8.6.1.1 How to represent modification of item shape?

The change section contains the modified items A, B, MH1 and KO1. In this case also the position of the instances relatively to the board needs to be described by another transformation, also the items “PCB” and ”board” are changed. Thus, they are also listed.

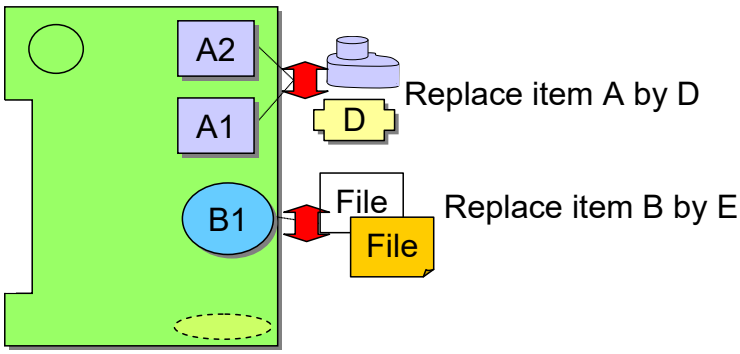
In the dataset section the new shape is described explicit by 2.5 D representation (see section 0) or implicit by a file (see section 7.1.9).

8.7 Test case 2-5: Replacement of items (Change 5)

Replace instances of PCB components (E.G. capacitor). This should cover following cases: shape of the component is represented implicit (by external file) or explicit (by 2.5 D geometry). Additionally, following cases should be covered: replace all instances of an item and only one of two (or more).

Figure 57 shows a possible constellation of test case 2-5.

Change 5 :



Situation after change 5 :

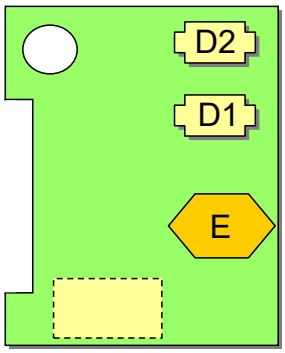


Figure 57: Verify replacement of items

8.7.1.1 How to represent replacement of items?

The change section lists the new items D, B, the item representing of the entire board (“PCB*”) as well as their predecessor items: (A, D), (B, E), (PCB, PCB*). The dataset section contains the (full) description of each new /modified item (D, E, PCB*).

9 Terms for IDX properties

This appendix summarises attribute names to be used within ECAD/MCAD collaboration.

In case the property describes a value of an item (E.g. CAPACITANCE of a capacitor), the IDX type *EDMDUserSimpleProperty* should be used. In case the property describes a constraint value set by a person, an *EDMDLogicProperty* (if value range is Boolean), *EDMDLengthProperty* (if max./min. length of an item is described) or *EDMDProperty* (other cases) should be used.

The following table shows the property name and unit to be used. This table has been created based on the terms used in IDF 3.0.

Note: this section remains for backward compatibility only.

Property	Type	Values	Unit	Comment
Thickness	float		-	**Specifies thickness of an item**
board side	string	TOP BOTTOM	-	
Routing layers	string	TOP	-	**Outline applies to top routing layer only**
		BOTTOM	-	
		BOTH	-	
		INNER	-	
		ALL	-	
Plating style	string	PTH	-	
		NPTH	-	
Associated part	string	BOARD	-	
		NOREFDES		
		PANEL	-	
		Reference	-	
		Designator	-	
Hole type	string	PIN	-	
		VIA	-	
		MTG	-	
		TOOL	-	
		Other	-	
Placement status	string	PLACED	-	
		UNPLACED	-	
		MCAD	-	

		ECAD		
Property name	float	CAPACITANCE	uF	**Capacitance, microfarads**
		RESISTANCE	Ohm	**Resistance, ohms**
		TOLERANCE	%	**Tolerance, % deviation**
		POWER_OPR	mW	**Operating power rating, Milliwatts**
		POWER_MAX	mW	**Maximum power rating, Milliwatts**
		THERM_COND	W/(mK)	**Thermal conductivity, Watt/meter_ C**
		THETA_JB	C/W	**Junction to board thermal resistance, C/Watt**
		THETA_JC	**C/W**	**Junction to case thermal resistance, C/Watt**

Table 11: Property names and units to be used (created on the terms used in IDF 3.0)

prostep IVIP

prostep ivip association

Dolivostraße 11
64293 Darmstadt
Germany

Phone +49-6151-9287336
Fax +49-6151-9287326

psev@prostep.com
www.prostep.org

ISBN 978-3-948988-00-5 v1.1
Version 4.5, 01/09/2020
PSI 35