Open EDA Component Library Format

Version 1.0

Important Information

Contents

Chapter 1 Introduction

About Open EDA Component Library 1-1
OECL Media Type, File Name Extension 1-1
  OECL namespace 1-1
Notation 1-1
Terminology 1-2

Chapter 2 Basic Datatypes

Definitions 2-1
Attribute Types 2-1
  The notEmptyString attribute 2-2
  The identifierType attribute 2-2
  The identifierRefType attribute 2-2
  The dateTimeRefType attribute 2-2
  The positiveDecimalType attribute 2-3
  The oeclVersionNumberType attribute 2-3
  The symbolMimeType attribute 2-3
  The symbolPinShapeType attribute 2-3
  The symbolPinLengthType attribute 2-3
  The placeholderType attribute 2-3
  The modelMimeType attribute 2-4
  The modelFormatType attribute 2-4
  The connectionNameType attribute 2-4
  The prefixStringType attribute 2-4
  The pinIdType attribute 2-4
  The symbolStyleType attribute 2-5
  The symbolStyleListType attribute 2-5
  The statusType attribute 2-5
  The availabilityType attribute 2-5
  The quantityAvailableType attribute 2-6
  The currencyType attribute 2-6
  The propertyKeyType attribute 2-6
  The propertyNameType attribute 2-6
  The measurementValueType attribute 2-6
  The layerType attribute 2-7
  The markingLayerType attribute 2-7
Element Types 2-7
  The Properties element 2-7
  The BooleanProperty element 2-8

The CurrencyProperty element 2-8
The DateTimeProperty element 2-9
The DoubleProperty element 2-9
The IntegerProperty element 2-10
The MeasurementProperty element 2-10
The TextProperty element 2-11
The URLProperty element 2-12
The Localization Element 2-12

Chapter 3 Document Structure

Defining an OECL Document 3-1
Overview 3-1
The root ComponentLibrary element 3-1

Chapter 4 Graphic symbols for schematics

The SymbolBlueprintDictionary element 4-1
The SymbolBlueprint element 4-1
The svg element 4-2
SVG for EDA symbols 4-2
Pin graphics 4-6
Placeholder text 4-8
Best practices 4-8
Example 1: resistor 4-8
Example 2: operational amplifier 4-9

Chapter 5 Simulation Models for SPICE Simulation

The SimulationModelBlueprintDictionary Element 5-1
The SimulationModelBlueprint element 5-1
The Instance element 5-2
The Definition element 5-3
Instance microsyntax 5-3
The Port element 5-4
The Variable element 5-4
Example 1 5-4

Chapter 6 Manufacturer Packages for Mapping to Standard Packages

The ManufacturerPackageDictionary element 6-1
The ManufacturerPackage element 6-1
The LandMap element 6-2
Example: DIP-ABC 6-2

Chapter 7 Packages for Printed Circuit Board Layout

The PackageBlueprintDictionary element 7-1

The PackageBlueprint element 7-1
The Package element 7-2
The SilkScreen element 7-3
The AssemblyDrawing element 7-3
The SolderMask element 7-4
The SolderPaste element 7-4
The Marking element 7-5
The Pin element 7-6
The Hole element 7-7
The Slot Element 7-8
Permitted elements 7-9
Example: DIP-6 7-9
Example: SOIC-8 7-11

Chapter 8 Category Information for Component Organization

The ComponentCategoryDictionary element 8-1
Component Category Dictionary Identifiers 8-1
The Category element 8-2
The LocalizedName element 8-2
The BlueprintRef element 8-3
The CategoryDictionaryRef element 8-4
Example 8-4

Chapter 9 Components for Representing Orderable Items

The ComponentBlueprintDictionary element 9-1
The ComponentBlueprint element 9-2
The Connections element 9-3
The Connection element 9-3
The Sections element 9-4
The Section element 9-4
The ConnInstance element 9-5
The SymbolConfigurations element 9-6
The SymbolConfiguration element 9-7
The Symbol element 9-7
The PinMap element 9-8
The SimulationModelConfigurations element 9-9
The SimulationModelConfiguration element 9-10
The SimulationModel element 9-10
The ModelPortMap element 9-11
The OrderablePackageConfigurations element 9-12
The OrderablePackageConfiguration element 9-12
The PadMap element 9-13
Example 1: resistor 9-14

Example 2: electrolytic capacitor 9-15
Example 3: electrolytic capacitor #2 9-15
Example 4: quad 2-input positive NAND gates 9-16

Chapter 10 Status Information for Component Obsolescence

The StatusDictionary element 10-1
The Status element 10-1
The Replacement element 10-2

Chapter 11 Order Information for Connections to Suppliers/Distributors

The OrderingInfoDictionary element 11-1
The OrderingInfo element 11-1
The Price Element 11-2
Example 11-3

Chapter 12 Standard Property Keys

Short Description 12-1
    Example 12-1
Description 12-1
    Example 12-1
Datasheet 12-1
    Example 12-1

Chapter 13 References

Chapter 1 Introduction

About Open EDA Component Library

This specification defines the features and syntax for the Open EDA Component Library (OECL).

OECL is an XML language for describing libraries for electronic design automation (EDA). Libraries include graphic symbols, simulation models, packages, land patterns, and components for the purpose of schematic capture, simulation, and printed circuit board layout. Components in this format can be imported into tools that support OECL.

OECL Media Type, File Name Extension

The media type (formerly known as MIME type) for OECL is "`application/oecl+xml`".

OECL files have the extension "`.oecl`" (all lower case) on all platforms.

OECL namespace

The following is the OECL 1.0 namespace:

```
http://www.oecl.org/2012/oecl
```

Notation

This document uses special fonts and coloring to identify XML syntax.

Item	Examples
Element	`ComponentLibrary`
Attribute name	`id`, `description`
Attribute value	`ANSI`, `true`
Schema type	`xsd:string`, `identifier`

Character data, CDATA	`<![CDATA[.model diodeexample`
	`+ IS=5e-6`
	`]]>`

Other	<, >, =""

Terminology

This section is non-normative.

This format uses attributes to define relationships between different objects. Where these attributes are intended to be displayed to the user, the attribute name generally has the suffix `Name`. Where these attributes are not intended to be displayed to the user, the attribute generally has the suffix `Id`.

Chapter 2 Basic Datatypes

OECL uses types defined in several standards. This specification uses the namespace prefixes in the following table to identify these types. Types may be used without namespace prefixes in OECL documents.

Datatypes	Definition	Prefix
W3C XML datatypes	http://www.w3.org/TR/2008/REC-xml-20081126/ (Extensible Markup Language (XML) 1.0 (Fifth Edition))	xml:
W3C XML schema datatypes	http://www.w3.org/2000/10/XMLSchema (XML Schema Part 2)	xsd:
W3C SVG datatypes	http://www.w3.org/TR/2011/REC-SVG11-20110816/	None
W3C XML Linking Language datatypes	http://www.w3.org/TR/xlink11 (XML Linking Language (XLink) Version 1.1)	xlink:
IPC-2581A	http://webstds.ipc.org/2581 (Generic Requirements for Printed Board Assembly Products Manufacturing Description Data and Transfer Methodology)	ipc:

OECL uses custom types defined in this specification. Custom types have no prefix.

Definitions

Property elements

One of the element types that define properties for a `ComponentBlueprint`. Specifically: `BooleanProperty`, `CurrencyProperty`, `DateTimeProperty`, `DoubleProperty`, `IntegerProperty`, `MeasurementProperty`, `TextProperty`, or `URLProperty`.

Core property attributes

Core property attributes are attributes that can be specified on all property elements. The core property attributes are `key` and `name`. Property elements must have a `key` and/or `name` attribute.

The `key` attribute assigns cross-implementation meaning to the property according to the name of the `key` (see Standard Property Keys). The value of the `key` attribute is not intended to be displayed. The `key` attribute type is `propertyKeyType`.

The `name` attribute is a display name for the property. The `name` attribute type is `propertyNameType`.

Attribute Types

The following section defines additional attribute basic types (SimpleTypes) for use in the OECL format.

The notEmptyString attribute

The `notEmptyString` type must contain at least one non-whitespace character. Whitespace characters are space, tab, line feed, and carriage return, and are identified by the `\s` XML Schema regular expression multi-character escape.

The identifierType attribute

The `identifierType` type is a universally unique identifier for an element. Identifier types always have a paired `xsd:dateTime` attribute. The combination of these two items should be universally unique. The `identifierType` type is derived from `notEmptyString`.

Constructing identifier attributes

This section is non-normative.

`identifierType` instances should be univerally unique, although there is no registration process. To ensure uniqueness, `identifierType` instances should be written in reverse domain name notation.

The reverse domain name notation should include a company/organization name, a category for the identifier, and a unique name for the item. Items that comply with a published standard, such as should include the standard body, a category for the identifier, and the unique name for the item.

Recommended categories for the identifiers are:

Element	Suggested category
`SymbolBlueprint`	`symb`
`ModelBlueprint`	`modl`
`ManufacturerPackage`	`mpkg`
`PackageBlueprint`	`pkg`
`ComponentBlueprint`	`comp`
`OrderablePackageConfiguration`	`ordpkg`

Examples of identifiers written using reverse domain name notation are:

```
com.domainname.symb.op-amp
com.companyname.comp.ABC123
org.ipc.pkg.BGA127P2X3-6
```

The identifierRefType attribute

The `identifierRefType` type is a reference to an `identifierType` attribute defined within the OECL document. The `identifierRefType` type is derived from `identifierType`.

The dateTimeRefType attribute

The `dateTimeRefType` type is a reference to an `xsd:dateTime` attribute defined within the OECL document. The `dateTimeRefType` type is derived from `xsd:dateTime`.

The positiveDecimalType attribute

The `positiveDecimalType` type defines a positive number. The `positiveDecimalType` type is derived from `xsd:double` by setting the value of `xsd:minInclusive` to `0`.

The oeclVersionNumberType attribute

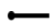
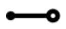
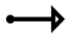
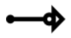
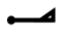
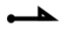
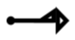
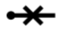
The `oeclVersionNumberType` type defines the format version of the document. The `oeclVersionNumberType` type is the single enumerated value `1.0`.

The symbolMimeType attribute

The `symbolMimeType` type identifies the format of the symbol information that follows. The `symbolMimeType` type is the single enumerated value `image/oecl.schsymbol+xml`.

The symbolPinShapeType attribute

The `symbolPinShapeType` type identifies the functional purpose of the pin. The `symbolPinShapeType` type is the set of enumerated values in the following table.

Enumerated value	Usage	Example (non-normative)
`passive`	The pin is a passive pin.	
`inverting`	The pin is an inverting pin.	
`clock`	The pin is a clock pin.	
`inverting-clock`	The pin is an inverting clock pin.	
`active-low`	The pin is an active low pin.	
`active-high`	The pin is an active high pin.	
`active-low-clock`	The pin is an active low clock pin.	
`nonlogic`	The pin is not a logical pin.	
`custom`	The shape is none of the other values. SVG elements describe the shape of the pin.	Not applicable

The symbolPinLengthType attribute

The `symbolPinLengthType` type identifies the length of the pin (without inspecting the drawing elements). The `symbolPinLengthType` type is one of the following: `0`, `10`, `20`, `30`, `40`.

The placeholderType attribute

The `placeholderType` type identifies the text that is substituted when a `SymbolBlueprint` is associated with a `ComponentBlueprint`. The `placeholderType` type is the set of enumerated values in the following table.

Enumerated value	Usage
`symbolPin`	The functional name of the pin.
`packagePad`	The name of the associated pad on the package.
`refDes`	The reference designator.
`properties`	The displayed list of component properties.

The modelMimeType attribute

The `modelMimeType` type identifies the format of the simulation model information that follows. The value `modelMimeType` type is the single enumerated value `application/oecl.spice+xml`.

The modelFormatType attribute

The `modelFormatType` type identifies compatible simulators for the simulation model. To determine compatible simulators, the value must be split on spaces. The resulting tokens identify compatible simulators for the model. Keywords are case-sensitive, and must not be specified more than once per attribute. A keyword beginning with a dash (-) explicitly identifies incompatible simulators for the model.

Splits on spaces indicate the “split a string on spaces algorithm” defined by HTML 5 (see 2.5.7 Space-separated tokens).

Keywords are implementation defined. Ignores unknown keywords.

Simulator Identifiers

This section is non-normative.

Keywords in the `modelFormatType` type should be globally unique, although there is no registration process. In order to ensure uniqueness, keywords should be written in reverse domain notation. Examples of reverse domain name notation are:

```
com.domainname.simulator
com.companyname
```

The connectionNameType attribute

The `connectionNameType` type identifies a name for a connection. The `connectionNameType` type must contain 1 to 8 characters from the set: a-z, A-Z, 0-9, +, -, *, ~.

The prefixStringType attribute

The `prefixStringType` type identifies the class designation letter portion of a reference designator, for example, as defined by IEC 81346. The `prefixStringType` is derived from `notEmptyString`.

The pinIdType attribute

The `pinIdType` type identifies the SVG element that names a symbol pin. The `pinIdType` is derived from `notEmptyString`.

The symbolStyleType attribute

The `symbolStyleType` type identifies standards that a symbol satisfies. The `symbolStyleType` type is the set of enumerated values in the following table.

Enumerated value	Usage
`IEEE_Std_315`	Complies with IEEE Std 315.
`IEC_60617`	Complies with IEC 60617.

The symbolStyleListType attribute

The `symbolStyleListType` type identifies standards that a symbol satisfied as a space-separated list. The `symbolStyleListType` type is derived from `xsd:list` by setting the value of `xsd:itemType` to `symbolStyleType`.

The statusType attribute

The `statusType` type identifies the availability of a component. The `statusType` type is the set of enumerated values in the following table.

Enumerated value	Usage
`available`	The component is in production and available.
`obsolete`	The component is obsolete.
`unknown`	Status information is unknown.
`other`	Status information is known, but not `available` or `obsolete`.

The availabilityType attribute

The `availabilityType` type identifies the availability of a component blueprint from a supplier/distributor. The `availabilityType` type is the set of enumerated values in the following table.

Enumerated value	Usage
`inStock`	The component is in stock.
`notStocked`	The component is not stocked.
`backOrder`	The component is on back order.
`RFQ`	Availability requires a request for quote.
`unknown`	Availability information is unknown.
`other`	Availability information is known, but not `inStock`, `notStocked`, `backOrder`, or `RFQ`.

The quantityAvailableType attribute

The `quantityAvailableType` type identifies the number of components in stock from a particular supplier/distributor. The `quantityAvailableType` type is the union of `xsd:nonNegativeInteger` and the enumerated value `unknown`.

The currencyType attribute

The `currencyType` type identifies a currency using the ISO 4217 3-character currency code. The `currencyType` type must contain 3 characters as defined by ISO 4217.

The following paragraph is non-normative.

OECL does not define specific three-character currency codes so that currencies can be added/removed from ISO 4217 without changing this specification. Applications should be designed to permit future addition of currency codes.

The propertyKeyType attribute

The `propertyKeyType` type identifies a property, where the meaning of the property is defined elsewhere. If the value is not recognized by the application, the application should handle the element as through the `name` attribute was specified. The `propertyKeyType` type is derived from `notEmptyString`.

The propertyNameType attribute

The `propertyNameType` type identifies a property, where the meaning of the property is defined by the attribute's content. The `propertyNameType` type is derived from `notEmptyString`.

The measurementValueType attribute

The `measurementValueType` type identifies a numeric value with optional SI unit prefix. The `measurementValueType` type is derived from `xsd:string` by setting the value of `xsd:pattern` to `[-+]?[0-9]*\.?[0-9]+([eE][-+]?[0-9]+)?([YZEPTGMkhdcmuµnpfazy]|da)?`.

The micro sign is the Unicode character U+00B5. In circumstances in which Unicode characters are not allowed, the micro prefix can be represented using the letter `u`.

The layerType attribute

The `layerType` type identifies the layer for a shape. The `layerType` type is the set of enumerated values in the following table.

Enumerated value	Usage
`all`	The shape is the same on all layers.
`top`	The shape is for the top layer of the PCB.
`inner`	The shape is for inner layers (not top or bottom) of the PCB.
`bottom`	The shape is for the bottom layer of the PCB.

The markingLayerType attribute

The `markingLayerType` type identifies the target layer for a shape. The `markingLayerType` type is the set of enumerated values in the following table.

Enumerated value	Usage
`top`	The shape is for the top layer of the PCB.
`bottom`	The shape is for the bottom layer of the PCB.

Element Types

The Properties element

`Properties`

Contexts in which this element can be used:

As an element in the `ComponentBlueprint` element

Content model:

Zero or more property elements

Attributes:

None

Defines properties for `ComponentBlueprint` instances.

The BooleanProperty element

`BooleanProperty`

Contexts in which this element can be used:

As an element in the `Properties` element

Content model:

Zero or more `Localization` elements

Attributes:

core property attributes

`value`

Defines a Boolean property.

Attribute	Attribute Type	Description	Required
`value`	`xsd:boolean`	The Boolean value	Yes

The CurrencyProperty element

`CurrencyProperty`

Contexts in which this element can be used:

As an element in the `Properties` element

Content model:

Zero or more `Localization` elements

Attributes:

core property attributes

`value`

`currency`

Defines a currency property.

Attribute	Attribute Type	Description	Required
`value`	`xsd:decimal`	The currency value	Yes
`currency`	`currencyType`	The currency using the ISO 4217 alphabetic code (3-character)	Yes

The DateTimeProperty element

`DateTimeProperty`

Contexts in which this element can be used:

As an element in the `Properties` element

Content model:

Zero or more `Localization` elements

Attributes:

core property attributes

`value`

Defines a date-time property.

Attribute	Attribute Type	Description	Required
`value`	`xsd:dateTime`	The date-time value	Yes

The DoubleProperty element

`DoubleProperty`

Contexts in which this element can be used:

As an element in the `Properties` element

Content model:

Zero or more `Localization` elements

Attributes:

core property attributes

`value`

Defines a numeric property.

Attribute	Attribute Type	Description	Required
`value`	`xsd:double`	The double value	Yes

The IntegerProperty element

`IntegerProperty`

Contexts in which this element can be used:

As an element in the `Properties` element

Content model:

Zero or more `Localization` elements

Attributes:

core property attributes

`value`

Defines an integer property.

Attribute	Attribute Type	Description	Required
`value`	`xsd:integer`	The integer value	Yes

The MeasurementProperty element

`MeasurementProperty`

Contexts in which this element can be used:

As an element in the `Properties` element

Content model:

Zero or more `Localization` elements

Attributes:

core property attributes

`value`

`unit`

Defines a measurable quantity property. Measurable quantities have values and units.

Attribute	Attribute Type	Description	Required
`value`	`measurementValueType`	The numeric value with optional SI unit prefix.	Yes
`unit`	`xsd:string`	The unit name or symbol.	Yes

The TextProperty element

`TextProperty`

Contexts in which this element can be used:

As an element in the `Properties` element

Content model:

Text content

Zero or more `Localization` elements

Attributes:

core property attributes

`value`

`xml:lang`

Defines a text property. Text properties may have a `value` attribute or text content, but not both. The `value` attribute and text content are equivalent.

Attribute	Attribute Type	Description	Required
`xml:lang`	`xml:language`	The language of the value. The initial value is implementation defined.	No
`value`	`xsd:string`	The text value or content.	Yes

The URLProperty element

`URLProperty`

Contexts in which this element can be used:

As an element in the `Properties` element

Content model:

Zero or more `Localization` elements

Attributes:

core property attributes

`value`

Defines a URL/hyperlink property.

Attribute	Attribute Type	Description	Required
`value`	`xsd:string`	The URL value.	Yes

The Localization Element

`Localization`

Contexts in which this element can be used:

As an element in one of the property elements

Content model:

Empty or text content

Attributes:

`xml:lang`

`name`

`value`

Defines a translated name or value for the parent element. Sibling elements must have unique values for `xml:lang`. Text content is permitted if the parent element is a `TextProperty`, otherwise, the content model is empty.

Attribute	Attribute Type	Description	Required
`xml:lang`	Language identifier as defined by IETF BCP 47.	The language of the translated value.	Yes
`name`	`notEmptyString`	Optional translated name for the property. Omit the attribute if the name is not translated.	No
`value`	`xsd:string`	Optional translated value for the property. Omit the attribute if the value is not translated or if the parent element is a `TextProperty` and the `Localization` element contains text content. A translated value is only permitted for the `TextProperty` and `URLProperty` elements.	No

Chapter 3 Document Structure

Defining an OECL Document

Overview

An OECL document consists of up to 8 dictionary elements contained within a `ComponentLibrary` element.

An OECL document can range from an empty fragment (that is, no content inside of the `ComponentLibrary` element), to individual dictionaries, to complex interdependent dictionaries.

An OECL document is a stand-alone, self-contained file or resource.

The root ComponentLibrary element

`ComponentLibrary`

Content model:

The following elements may appear in any order

Zero or one `SymbolBlueprintDictionary`

Zero or one `SimulationModelBlueprintDictionary`

Zero or one `ManufacturerPackageDictionary`

Zero or one `PackageBlueprintDictionary`

Zero or one `ComponentCategoryDictionary`

Zero or one `ComponentBlueprintDictionary`

Zero or one `StatusDictionary`

Zero or one `OrderingInfoDictionary`

Attributes:

`version`

The `ComponentLibrary` element represents the root of an OECL document.

Attribute Name	Attribute Type	Description	Required
`version`	`oeclVersionNumberType`	Indicates the OECL language version to which this document conforms. This attribute is fixed to the value of `1.0`.	Yes

Chapter 4 Graphic symbols for schematics

The SymbolBlueprintDictionary element

`SymbolBlueprintDictionary`

Contexts in which this element can be used:

As an element in the `ComponentLibrary` element

Content model:

Zero or more `SymbolBlueprint` elements

Attributes:

None

Defines reusable graphic symbols for schematic capture. Each reusable graphic symbol is described by a `SymbolBlueprint` element.

The SymbolBlueprint element

`SymbolBlueprint`

Contexts in which this element can be used:

As an element in the `SymbolBlueprintDictionary` element

Content model:

One `svg` element

Attributes:

`id`

`mimetype`

`description`

`revisionDate`

Each graphic symbol is composed of two types of information: connection points (pins) for schematic capture wiring and primitive graphic elements for drawing the symbol.

Attribute Name	Attribute Type	Description	Required
`id`	`identifier`	Unique identifier for the graphic symbol.	Yes
`mimetype`	`symbolMimeType`	Identifies the format of the sub-element symbol information. This attribute is fixed to the value `image/oecl.schsymbol+xml`.	Yes
`description`	`xsd:string`	Description of the symbol graphic for identifying the meaning of the graphic symbol. The initial value is a zero-length string.	No
`revisionDate`	`xsd:dateTime`	Date of this revision.	Yes

The svg element

Although the symbol is described using SVG, OECL uses a restricted sub-set of elements and attributes that is appropriate for electronics schematics.

A graphic symbol for schematic capture contains graphical elements (for the purpose of drawing) and connection elements (for the purpose of wiring). Connection elements are handled through extensions to the SVG format.

SVG for EDA symbols

SVG for EDA symbols consist of the following SVG 1.1 modules. For each module, the permissible attributes/elements in each module are given following the module name. Additional restrictions that apply to coordinate values, presentation attribute values, and the transform attribute are described in Additional restrictions.

In addition, SVG for EDA symbols defines additional attributes for the purpose of describing pins and substitution text. The following sections define these attributes.

Structure

Core attribute module

Collection name	Attributes in Collection
Core.attrib	id

Structure module

Element	Attributes	Content Model
`defs`	`Core.attrib, transform`	`(Structure.class | Shape.class | View.class | Text.class)*`
`g`	`Core.attrib, transform, Paint.attrib, OeclConn.attrib`	`(Structure.class | Shape.class | View.class | Text.class)*`
`svg`	`Core.attrib, x, y, width, height, viewBox, version`	`(Structure.class | Shape.class | Image.class | Text.class)*`
`use`	`Core.attrib, Paint.attrib, XLink.attrib, OeclConnType.attrib, transform, x, y, width, height`	`Empty`

Painting module

Basic paint attribute module

Collection name	Attributes in Collection
`Paint.attrib`	`fill`, `stroke`, `stroke-width`, `opacity`

Property inheritance is not supported.

Basic graphics attribute module

Collection name	Attributes in Collection
`Graphics.attrib`	`visibility`

Property inheritance is not supported.

Hyperlinking

XLink attribute module

Collection name	Attributes in Collection
`XLink.attrib`	`xlink:href`

Shapes

Shapes module

Element	Attributes	Content Model
`circle`	`Core.attrib, Paint.attrib, Graphics.attrib, OeclConnPoint.attrib, cx, cy, r, transform`	`Animation.class`
`ellipse`	`Core.attrib, Paint.attrib, Graphics.attrib, cx, cy, rx, ry, transform`	`Animation.class`
`line`	`Core.attrib, Paint.attrib, Graphics.attrib, x1, y1, x2, y2, transform`	`Animation.class`
`path`	`Core.attrib, Paint.attrib, Graphics.attrib, d, transformd`	`Animation.class`
`polygon`	`Core.attrib, Paint.attrib, Graphics.attrib, points, transform`	`Animation.class`
`polyline`	`Core.attrib, Paint.attrib, Graphics.attrib, points, transform`	`Animation.class`
`rect`	`Core.attrib, Paint.attrib, Graphics.attrib, x, y, width, height, rx, ry, transform`	`Animation.class`

Text

Basic text module

Element	Attributes	Content Model
`text`	`Core.attrib, Paint.attrib, Graphics.attrib, Font.attrib, TextContent.attrib, OeclLabel.attrib, transform, x, y`	`#PCDATA`

`x` and `y` attributes must be a single-coordinate or not specified.

Basic text attribute set

Collection name	Attributes in Collection
`Font.attrib`	`font-family`, `font-size`, `font-style`, `font-weight`
`TextContent.attrib`	`text-anchor`

font-family attribute

A single <family-name> is permitted.

Value: <family-name>

font-size attribute

Font sizes must be specified in px. The px unit is not required.

Value: <absolute-size>

font-style attribute

The following subset of values is permitted.

Value: `normal` | `italic`

Implementations should interpret other values as `normal`.

font-weight

The following subset of values is permitted.

Value: `normal` | `bold`

Implementations should interpret other values as `normal`.

Additional restrictions

SVG for EDA symbols must satisfy the below additional restrictions.

All coordinate values must be unit-less.

The `inherit` attribute value is not allowed for presentation attributes.

The `transform` attribute permits only combinations of the following transforms:

- Rotate in 90 degree increments
- Scaling
- Translation

OECL Extension Module

OECL Attribute Set

Collection name	Attributes in Collection
`OeclConn.attrib`	`oecl:pinId`
`OeclConnPoint.attrib`	`oecl:connPoint`
`OeclLabel.attrib`	`oecl:placeholder`
`OeclConnType.attrib`	`oecl:pin`, `oecl:pinLength`

Attribute	Attribute Type	Description
`oecl:pinId`	`pinIdType`	Identifier for the symbol pin. This ID is used as a reference from a `ComponentBlueprint`.
`oecl:connPoint`	`xsd:boolean`	Identifies the location of the connection point for wiring. Required on a circle element that defines the location of the connection point. Not allowed in other contexts.
`oecl:placeholder`	`placeholderType`	Text elements with this attribute define placeholder text items that are substituted with values from a `ComponentBlueprint` when placed on a schematic.
`oecl:pin`	`symbolPinShapeType`	Enumerated shape of the pin. Required on `use` elements that define a pin. Not allowed in other contexts.
`oecl:pinLength`	`symbolPinLengthType`	Enumerated length of the pin. Required on `use` elements that define a pin. Not allowed in other contexts.

Pin graphics

Symbols contain pins which form the connection points to the symbol and display information about the connection point. Pins have standard types, and these standard types are instantiated by the graphic. Functionally, instances of a pin contain:

- an identifier for the pin, for reference by component blueprints, and
- a reference to the standard pin type and default shape, and
- placeholder text elements that are substituted when the symbol is referenced by a component blueprint.

Instances of a pin are described by a `g` (group) with the `oecl:pinId` attribute element that contains the identifier for the pin. `g` elements with the `oecl:pinId` attribute must define an instance of a pin.

`g` elements that define a pin instance contain three sub-elements:

- one `use` element that identifies the shape of the pin.
- one `text` element that identifies substitution text for the symbol pin name from a referencing component blueprint.
- one `text` element that identified substitution text for the package pad from a referencing component blueprint.

The substitution text elements may be omitted.

The use element for pin graphics

The `use` element with the `oecl:pin` attribute defines the graphic for the pin. The element references a graphic contained in the `defs` section that provides a default graphic for the pin. This information is normally useful for viewing the graphic in a standard SVG viewer.

The `use` element must contain the `oecl:pin` attribute. The `oecl:pin` attribute describes the functional purpose of the pin.

The `x` and `y` attributes of the `use` element must be located on a grid of 10 in the initial viewport coordinate system. See Drawing pin graphics for additional information.

The following paragraph is non-normative.

Implementations typically use the `oecl:pin` attribute to identify the functional purpose of the pin. Implementations typically ignore the graphic objects referenced by the `xlink:href` attribute.

The text element for pin graphics

The `text` element with the `oecl:placeholder` element defines placeholder text elements used when a symbol blueprint is referenced by a component blueprint. Each pin defines two placeholder text elements, one for the connection name (see the `PinMap` element) and one for the package pad name (see the `PadMap` element).

Implementations replace the content of the `text` element with the appropriate name from the referencing component blueprint. The appropriate name is determined by the `oecl:placeholder` attribute, and described by the following table.

Enumerated value	**Usage**
`symbolPin`	The functional name of the pin.
`packagePad`	The name of the associated pad on the package.

Drawing pin graphics

Connection points for wiring must be drawn on a grid of 10. That is, 10 is defined as the implementation-specific minimum Manhattan distance between pins.

Pin length must be 0, 10, 20, 30, or 40. The connection point for schematic wiring for all pins is at (0,0). For pins with a primary line segment, the line segment ends at (0,0). The pin starts at (0,0), (10,0), (20,0), (30,0), or (40,0), depending on the length of the pin. Enumerated values for the `oecl:pinLength` attribute define possible pin lengths. The length of the referenced graphic must equal the length of the `oecl:pinLength` attribute.

Enumerated values for the `oecl:pin` attribute define possible pin types.

The `passive` pin graphics can have lengths of 0, 10, 20 or 40. Other pin graphics can have lengths of 10, 20, 30, or 40.

The `custom` `oecl:pin` type can be drawn either as specified by the SVG drawing objects or as the `passive` `oecl:pin` type.

The following paragraph is non-normative

Implementations typically use the `oecl:pin` attribute to identify the functional purpose of the pin. Implementations typically ignore the graphic objects referenced by the `xlink:href` attribute. For readability, implementations should name the referenced `id` to match the standard pin type and length, separated by a dash. For example, the `id` of the passive pin type with length 10 should be `passive-10` and the `xlink:href` should be `#passive-10`.

Placeholder text

Symbols may contain placeholder `text` elements whose value is replaced with information from the referencing component blueprint, for example the component name, or information from the schematic, for example the reference designator.

`text` elements with the `oecl:placeholder` attribute identify placeholder text, as described by the following table.

Enumerated value	Usage
`refDes`	The reference designator.
`properties`	The displayed list of component properties.

Implementations may ignore text elements with the `oecl:placeholder` attribute.

The following paragraph is non-normative

Implementations typically ignore the text elements with the `oecl:placeholder` attribute. These elements are typically used for displaying the graphic symbol with no connection to a schematic.

Best practices

This section is non-normative

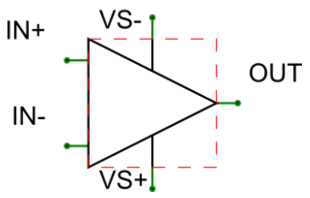
Implementations typically have implementation-specific styles for drawing components for colors and font that provide a consistent look for the application. Symbols should typically omit this style information and allow the implementation to provide implementation-specific default styles.

Shapes should usually omit the `fill` and `stroke` attributes (except the value `none`) so that implementations may use their default color information.

Text should usually omit the font attributes (Font.attrib) so that implementations may use their default font information.

Many implementations require all pins to originate from the boundary of a box enclosing the symbol contents. For best results across multiple applications, pins should be on a bounding box enclosing the symbol.

The following graphic illustrates this concept. The pins (green), end on the boundary of the drawing objects (red). The pins labeled VS- and VS+ have additional drawing objects to connect the pin to triangle.



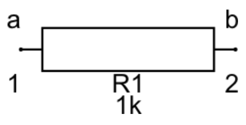
Example 1: resistor

The following XML fragment defines the symbol blueprint for the IEC 60617 resistor (S00555).

```
<SymbolBlueprint id="com.example.symb.resistor" mimetype="image/oecl.schsymbol+xml"
description="IEC 60617 S00555 Resistor, general symbol" revisionDate="2012-07-17T21:32:52">
    <svg xmlns:svg="http://www.w3.org/2000/svg"
```

```
        xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:oecl="http://www.oecl.org/2012/oecl"
        xmlns="http://www.w3.org/2000/svg">
        <!--Common pin definitions-->
        <defs>
            <g id="passive-10">
                <line x1="10" y1="0" x2="0" y2="0"/>
                <!--The following circle with the connPoint attribute defines
                the connection point for the pin-->
                <circle cx="0" cy="0" r="1" oecl:connPoint="true"/>
            </g>
        </defs>
        <!--Symbol pins-->
        <g oecl:pinId="1">
            <use transform="translate(-50,0)" xlink:href="#passive-10" oecl:pin="passive"
oecl:pinLength="10"/>
            <!--Placeholder text for labels on the symbol-->
            <text transform="translate(-50,-10)" text-anchor="end"
oecl:placeholder="symbolPin">a</text>
            <text transform="translate(-50,20)" text-anchor="end"
oecl:placeholder="packagePad">1</text>
        </g>
        <g oecl:pinId="2">
            <use transform="translate(50,0) rotate(180)" xlink:href="#passive-10"
oecl:pin="passive" oecl:pinLength="10"/>
            <!--Substitution text for labels on the symbol-->
            <text transform="translate(45,-10)" oecl:placeholder="symbolPin">b</text>
            <text transform="translate(45,20)" oecl:placeholder="packagePad">2</text>
        </g>
        <!--Graphic symbol-->
        <rect x="-40" y="-10" width="80" height="20" fill="none"/>
        <text x="0" y="20" oecl:placeholder="refDes" text-anchor="middle">R1</text>
        <text x="0" y="30" oecl:placeholder="properties" text-anchor="middle">1k</text>
    </svg>
</SymbolBlueprint>
```

Graphically, this XML fragment draws as shown below.



Example 2: operational amplifier

```
<svg xmlns:svg="http://www.w3.org/2000/svg"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:oecl="http://www.oecl.org/2012/oecl"
    xmlns="http://www.w3.org/2000/svg">
    <!--Common pin definitions-->
    <defs>
        <g id="passive-10">
```

```
            <line x1="0" y1="0" x2="10" y2="0"/>
            <!--The following circle with the connPoint attribute defines
            the connection point for the pin-->
            <circle cx="0" cy="0" r="1" oecl:connPoint="true"/>
        </g>
    </defs>
    <!--Symbol pins-->
    <g oecl:pinId="IN+">
        <use transform="translate(-40,-20)" xlink:href="#passive-10" oecl:pin="passive"
oecl:pinLength="10"/>
        <!--Placeholder text for labels on the symbol-->
        <text transform="translate(-50,-30)" text-anchor="end"
oecl:placeholder="symbolPin">IN+</text>
    </g>
    <g oecl:pinId="IN-">
        <use transform="translate(-40,20)" xlink:href="#passive-10" oecl:pin="passive"
oecl:pinLength="10"/>
        <!--Placeholder text for labels on the symbol-->
        <text transform="translate(-50,10)" text-anchor="end"
oecl:placeholder="symbolPin">IN-</text>
    </g>
    <g oecl:pinId="OUT">
        <use transform="translate(40,0) rotate(180)" xlink:href="#passive-10"
oecl:pin="passive" oecl:pinLength="10"/>
        <!--Substitution text for labels on the symbol-->
        <text transform="translate(45,-10)" oecl:placeholder="symbolPin">OUT</text>
    </g>
    <g oecl:pinId="VS-">
        <use transform="translate(0,40) rotate(-90)" xlink:href="#passive-10"
oecl:pin="passive" oecl:pinLength="10"/>
        <!--Substitution text for labels on the symbol-->
        <text transform="translate(-25,40)" oecl:placeholder="symbolPin">VS-</text>
    </g>
    <g oecl:pinId="VS+">
        <use transform="translate(0,-40) rotate(90)" xlink:href="#passive-10"
oecl:pin="passive" oecl:pinLength="10"/>
        <!--Substitution text for labels on the symbol-->
        <text transform="translate(-25,-35)" oecl:placeholder="symbolPin">VS+</text>
    </g>
    <!--Graphic symbol-->
    <path d="M -30 -30 L 30 0 L -30 30 z"/>
    <line x1="0" y1="15" x2="0" y2="30"/>
    <line x1="0" y1="-15" x2="0" y2="-30"/>
</svg>
```

Chapter 5 Simulation Models for SPICE Simulation

The SimulationModelBlueprintDictionary Element

SimulationModelBlueprintDictionary

Contexts in which this element can be used:

As an element in the ComponentLibrary element

Content model:

Zero or more SimulationModelBlueprint elements

Attributes:

None

Defines reusable models for simulation. Each reusable simulation model is described by a SimulationModelBlueprint element.

The SimulationModelBlueprint element

SimulationModelBlueprint

Contexts in which this element can be used:

As an element in the SimulationModelBlueprintDictionary element

Content model:

One or more Instance elements followed by one or more Definition elements

Attributes:

id

name

manufacturer

author

mimetype

description

copyright

revisionDate

There are two parts to the model blueprint, (1) instance information and (2) model text, represented by the Instance and Model elements, respectively. The instance information is used to instantiate the model (make the model specific to a particular component), whereas the model text is common (shared) between instances of the model.

Attribute	Attribute Type	Description	Required
`id`	`identifierType`	Unique identifier for the simulation model.	Yes
`name`	`notEmptyString`	Descriptive name of the model. For SPICE models, this is the name of the `.subckt` or `.model`.	Yes
`manufacturer`	`notEmptyString`	The name of the manufacturer for models that describe the behavior from a particular manufacturer. The initial value is a zero-length string.	No
`author`	`xsd:string`	Name of the model publisher. The initial value is a zero-length string.	No
`mimetype`	`modelMimeType`	Identifies the format of the sub-element model information.	Yes
`description`	`xsd::string`	Description information for the model.	No
`copyright`	`xsd:string`	Model copyright information. The initial value is a zero-length string.	No
`revisionDate`	`xsd:dateTime`	Date of this revision.	Yes

The Instance element

`Instance`

Contexts in which this element can be used:

As an element in the `SimulationModelBlueprint` element

Content model:

Instance text in a CDATA section

Attributes:

`simulators`

Describes how to instantiate the simulation model (that is, map ports).

Attribute	Attribute Type	Description	Required
`simulators`	`modelFormatType`	Identifies compatible simulators for the model. The missing or empty attribute indicates unknown information. Listed values must be unique for all `Instance` elements of the same `SimulationModelBlueprint`.	No

The Definition element

`Definition`

Contexts in which this element can be used:

As an element in the `SimulationModelBlueprint` element

Content model:

Model text or encrypted model text in a CDATA section

Attributes:

`simulators`

Shared information for a simulation model. The name of the SPICE `.subckt` or `.model` must match the value of the `SimulationModelBlueprint`'s `name` attribute.

Attribute	Attribute Type	Description	Required
`simulators`	`modelFormatType`	Identifies compatible simulators for the model. The missing or empty attribute indicates unknown information. Listed values must be unique for all `Definition` elements of the same `SimulationModelBlueprint`.	No

Instance microsyntax

The CDATA section for the `Instance` element describes how to instantiate the SPICE model.

The syntax is an XML-like markup language where all characters, including white space, is significant. The syntax does not have a root element. The CDATA section for the `Instance` element may contain the following elements in any order:

- text content block
- `Port` element
- `Variable` element

The Port element

Port

Contexts in which this element can be used:

As an element in the `Instance` CDATA section element

Content model:

One text content block. The text content block contains only the name of the terminal

Attributes:

`behavior`

Defines the name of a port (terminal).

Attribute	Attribute Type	Description	Required
`behavior`	`xsd:string`	Simulator-specific information that describes the electrical behavior of the port. The initial value is a zero-length string.	No

The Variable element

Variable

Contexts in which this element can be used:

As an element in the `Instance` CDATA section element

Content model:

The string `REFDES` or the string `MODEL`

Attributes:

None

Identifies replacement text for the reference designator of the instantiated model or the name of the instantiated model.

Example 1

The following XML fragment defines the model blueprint for a 1 kΩ resistor.

```
<SimulationModelBlueprint id="BBC45068-2D72-4385-8E1C-A77D7E695F13" name="myres"
manufacturer="Generic" mimetype=" application/oecl.spice+xml" revisionDate="2012-07-
17T21:32:52">
    <Instance><![CDATA[x<Variable>REFDES</Variable> <Port>T0</Port> <Port>T1</Port>
<Variable>MODEL</Variable>]]></Instance>
    <Definition><![CDATA[.subckt myres 1 2
R1 1 2 1k
.ends]]></Definition>
```

```
</SimulationModelBlueprint>
```

Chapter 6 Manufacturer Packages for Mapping to Standard Packages

The ManufacturerPackageDictionary element

`ManufacturerPackageDictionary`

Contexts in which this element can be used:

As an element in the `ComponentLibrary` element

Content model:

Zero or more `ManufacturerPackage` elements

Attributes:

None

Provides definitions of manufacturer-specific packages names/pin numbering and mapping of the package to reusable standard packages. Each manufacturer-specific package is described by a `ManufactuerPackage` element.

The ManufacturerPackage element

`ManufacturerPackage`

Contexts in which this element can be used:

As an element in the `ManufacturerPackageDictionary` element

Content model:

One or more `LandMap` elements

Attributes:

`id`

`name`

`manufacturer`

`packageBlueprintRef`

`packageBlueprintdateTime`

`revisionDate`

Defines the manufacturer-specific name for a package and mapping to the standard package.

Attribute	Attribute Type	Description	Required
`id`	`identifierType`	Unique identifier for the manufacturer's package.	Yes
`name`	`notEmptyString`	Manufacturer name of the package.	Yes
`manufacturer`	`notEmptyString`	Manufacturer name. The initial value is a zero-length string.	No
`packageBlueprintRef`	`identifierRefType`	Unique identifier for the associated package.	Yes
`packageBlueprintDateTime`	`xsd:dateTime`	The `dateTime` of the referenced `PackageBlueprint`. If omitted, matches the most recent `PackageBlueprint`.	No
`revisionDate`	`xsd:dateTime`	Date of this revision.	Yes

The LandMap element

`LandMap`

Contexts in which this element can be used:

As an element in the `ManufacturerPackage` element

Content model:

Empty

Attributes:

`padName`

`landName`

Defines how pad names on the manufacturer package map to land names on the standard package.

Attribute	Attribute Type	Description	Required
`padName`	`notEmptyString`	Manufacturer name of the pad/electrical connection point.	Yes
`landName`	`notEmptyString`	The name of the land on the referenced package blueprint.	Yes

Example: DIP-ABC

The following XML fragment defines a manufacturer package.

```
<ManufacturerPackage id="com.example.mpkg.DIP-ABC" name="DIP-ABC" manufacturer="Generic
Manufacturer" packageBlueprintRef="com.example.pkg.DIP-8" revisionDate="2012-07-17T21:32:52">
    <LandMap padName="A" landName="1"/>
    <LandMap padName="B" landName="2"/>
</ManufacturerPackage>
```

Chapter 7 Packages for Printed Circuit Board Layout

The PackageBlueprintDictionary element

PackageBlueprintDictionary

Contexts in which this element can be used:

As an element in the ComponentLibrary element

Content model:

Zero or more PackageBlueprint elements

Attributes:

None

Defines reusable packages (footprints, land-patterns) for components. Each reusable package is described by a PackageBlueprint element. For example, surface mount land patterns generally should comply with IPC-7351 Generic Requirements for Surface Mount Design and Land Pattern Standard.

The PackageBlueprint element

PackageBlueprint

Contexts in which this element can be used:

As an element in the PackageBlueprintDictionary element

Content model:

One Package element

Attributes:

id

name

revisionDate

units

Defines a package (which includes the land pattern).

Attribute	Attribute Type	Description	Required
id	identifierType	Unique identifier for the package.	Yes
name	notEmptyString	A unique name for the package. Typically, this name follows the IPC-7351 series of part and land pattern descriptions.	Yes
revisionDate	xsd:dateTime	Date of this revision.	Yes
units	ipc:unitsType	An enumerated string that must be one of the following: MILLIMETER | MICRON | INCH.	Yes

The Package element

Package

Contexts in which this element can be used:

As an element in the PackageBlueprint element

Content model:

One Outline element (defined as ipc:OutlineType)

Zero or one SilkScreen element

Zero or one AssemblyDrawing element

Zero or one SolderMask element

Zero or one SolderPaste element

Zero or more Pin elements

Attributes:

name

type

pinOne

height

Defines a package (which includes the land pattern). The package elements are based on a modified version of IPC-2581A. Elements that are identical to elements defined in IPC-2581A are not listed in this specification.

The following paragraph is non-normative.

The value of the name attribute is often the same as the value of the name attribute of the parent element.

Attribute	Attribute Type	Description	Required
name	ipc:qualifiedNameType	See IPC-2581A.	Yes
type	ipc:PackageTypeType	See IPC-2581A.	Yes
pinOne	xsd:string	See IPC-2581A.	No
height	xsd:double	See IPC-2581A.	No

The SilkScreen element

SilkScreen

Contexts in which this element can be used:

As an element in the Package element

Content model:

Zero or more Outline elements (defined as ipc:OutlineType)

Zero or more Marking elements

Attributes:

None

The SilkScreen element is based on the ipc:SilkScreenType from IPC-2581A.

The AssemblyDrawing element

AssemblyDrawing

Contexts in which this element can be used:

As an element in the Package element

Content model:

Zero or one Outline elements (defined as ipc:OutlineType)

Zero or more Marking elements

Attributes:

None

The AssemblyDrawing element is based on the ipc:AssemblyDrawingType from IPC-2581A.

The SolderMask element

`SolderMask`

Contexts in which this element can be used:

As an element in the `Package` element

Content model:

Zero or one `Outline` elements (defined as `ipc:OutlineType`)

Zero or more `Marking` elements

Attributes:

None

The `SolderMask` element reuses the same embedded elements and attributes as defined for the `SilkScreen` element. If omitted, all lands on the package have solder mask that matches the shape of the land.

The SolderPaste element

`SolderPaste`

Contexts in which this element can be used:

As an element in the `Package` element

Content model:

Zero or one `Outline` elements (defined as `ipc:OutlineType`)

Zero or more `Marking` elements

Attributes:

None

The `SolderPaste` element reuses the same embedded elements and attributes as defined for the `SilkScreen` element. If omitted, all lands on the package have solder paste that matches the shape of the land.

The Marking element

Marking

Contexts in which this element can be used:

As an element in the SilkScreen, AssemblyDrawing, SolderMask, or SolderPaste elements

Content model:

Zero or one Xform elements (defined as ipc:XformType)

One Location element (defined as ipc:LocationType)

Zero or more Feature elements (ABSTRACT type defined in IPC-2581A)

Attributes:

markingUsage

layer

The Marking element is based on the ipc:markingType from IPC-2581A with the addition of the layer attribute.

Attribute	Attribute Type	Description	Required
markingUsage	ipc:markingUsageType	See IPC-2581A.	Yes
layer	markingLayerType	Defines the layer for the marking. The initial value is top.	No

The Pin element

Pin

Contexts in which this element can be used:

As an element in the Package element

Content model:

Zero or one Xform elements (defined as ipc:XformType)

One Location element (defined as ipc:LocationType)

One or three StandardShape elements (ABSTRACT type defined in IPC-2581A). See below for additional attributes on the StandardShape element

Zero or one Hole or Slot elements

Attributes:

number

name

type

electricalType

mountType

The Pin element is based on the ipc:PinType from IPC-2581A. The Pin element defines the pad stack and contains either one StandardShape substitution group child element or three StandardShape substitution group child elements.

For through-hole pins (type is THRU or BLIND), if the Pin element contains one StandardShape substitution group child element, the pad is the same on all layers. If the Pin element contains three StandardShape substitution group child elements, each StandardShape substitution group child element must contain a unique oecl:pinLayer attribute (type oecl:layerType), and the pad shape is as specified on the layer.

For through-hole pins (type is THRU or BLIND), one Hole or Slot element is required and the element defines the shape of the formed hole for the pin.

For surface mount pads (type is SURFACE), the Hole or Slot element is not allowed.

Attribute	Attribute Type	Description	Required
number	ipc:qualifiedNameType	See IPC-2581A.	Yes
name	ipc:qualifiedNameType	See IPC-2581A.	No
type	ipc:cadPinType	See IPC-2581A.	Yes
electricalType	ipc:pinElectricalType	See IPC-2581A.	No
mountType	ipc:pinMountType	See IPC-2581A.	No

The Hole element

Hole

Contexts in which this element can be used:

As an element in the `Pin` element

Content model:

Empty

Attributes:

`name`

`diameter`

`platingStatus`

`plusTol`

`minusTol`

`x`

`y`

The `Hole` element is based on the `ipc:HoleType` from IPC-2581A. The `Hole` element defines a circular drill hole associated with a through-hole pin.

The `x` and `y` attributes offset the hole relative to the `Pin` and are typically 0.

Attribute	Attribute Type	Description	Required
`name`	`ipc:qualifiedNameType`	See IPC-2581A.	Yes
`diameter`	`ipc:nonNegativeDoubleType`	See IPC-2581A.	Yes
`platingStatus`	`ipc:platingStatusType`	See IPC-2581A.	Yes
`plusTol`	`ipc:nonNegativeDoubleType`	See IPC-2581A.	Yes
`minusTol`	`ipc:nonNegativeDoubleType`	See IPC-2581A.	Yes
`x`	`xsd:double`	See IPC-2581A.	Yes
`y`	`xsd:double`	See IPC-2581A.	Yes

The Slot Element

`Slot`

Contexts in which this element can be used:

As an element in the `Pin` element

Content model:

One or more `ipc:Arc`, `ipc:Line`, `ipc:Outline` and/or `ipc:Polyline` elements

Attributes:

`name`

`platingStatus`

`plusTol`

`minusTol`

The `Slot` element is based on the `ipc:SlotType` from IPC-2581A. The `Slot` element defines an arbitrarily shaped drill hole associated with a through-hole pin.

The positions of the child elements are relative to the `Pin`.

Attribute	Attribute Type	Description	Required
`name`	`ipc:qualifiedNameType`	See IPC-2581A.	Yes
`platingStatus`	`ipc:platingStatusType`	See IPC-2581A.	Yes
`plusTol`	`ipc:nonNegativeDoubleType`	See IPC-2581A.	Yes
`minusTol`	`ipc:nonNegativeDoubleType`	See IPC-2581A.	Yes

Permitted elements

The content model of the `PackageBlueprint` element contains one `Package` element. All child elements defined by IPC-2581A for the `Package` element are permitted, except the following:

```
LineDescRef
StandardPrimitiveRef
UserPrimitiveRef
```

Example: DIP-6

The following XML fragment defines a package blueprint the DIP-6 package.

```
<PackageBlueprint id="DIP762W46P254L876Q6B" name="DIP-6" revisionDate="2012-07-17T21:32:52"
units="MILLIMETER">
    <Package name="DIP-6" type="CERAMIC_DIP">
        <Outline>
            <LineDesc lineEnd="ROUND" lineWidth="0.001"/>
            <Polygon>
                <PolyBegin x="-4.635" y="4.945"/>
                <PolyStepSegment x="-4.635" y="-4.945"/>
                <PolyStepSegment x="4.635" y="-4.945"/>
                <PolyStepSegment x="4.635" y="4.945"/>
                <PolyStepSegment x="-4.635" y="4.945"/>
            </Polygon>
        </Outline>
        <SilkScreen>
            <Outline>
                <LineDesc lineEnd="ROUND" lineWidth="0.2"/>
                <Polygon>
                    <PolyBegin x="-2.9" y="4.4"/>
                    <PolyStepSegment x="-2.9" y="-4.4"/>
                    <PolyStepSegment x="2.9" y="-4.4"/>
                    <PolyStepSegment x="2.9" y="4.4"/>
                    <PolyStepSegment x="-2.9" y="4.4"/>
                </Polygon>
            </Outline>
            <Marking markingUsage="PIN_ONE">
                <Location x="-1.88" y="3.4"/>
                <Circle diameter="1.0"/>
            </Marking>
        </SilkScreen>
        <AssemblyDrawing>
```

```
            <Outline>
                <LineDesc lineEnd="ROUND" lineWidth="0.1"/>
                <Polygon>
                    <PolyBegin x="-3.21" y="4.38"/>
                    <PolyStepSegment x="-3.21" y="-4.38"/>
                    <PolyStepSegment x="3.21" y="-4.38"/>
                    <PolyStepSegment x="3.21" y="4.38"/>
                    <PolyStepSegment x="-3.21" y="4.38"/>
                </Polygon>
            </Outline>
            <Marking markingUsage="PIN_ONE">
                <Location x="-2.2" y="3.37"/>
                <Donut shape="ROUND" outerDiameter="1.0" innerDiameter="0.8"/>
            </Marking>
        </AssemblyDrawing>
        <Pin number="1" type="THRU" electricalType="ELECTRICAL" mountType="THROUGH_HOLE_PIN">
            <Location x="-3.81" y="2.54"/>
            <RectCenter width="1.15" height="1.15"/>
            <Hole name="1" diameter="0.5" platingStatus="PLATED" plusTol="0.005"
minusTol="0.005" x="0" y="0"/>
            </Pin>
        <Pin number="2" type="THRU" electricalType="ELECTRICAL" mountType="THROUGH_HOLE_PIN">
            <Location x="-3.81" y="0"/>
            <Circle diameter="1.15"/>
            <Hole name="2" diameter="0.5" platingStatus="PLATED" plusTol="0.005"
minusTol="0.005" x="0" y="0"/>
        </Pin>
        <Pin number="3" type="THRU" electricalType="ELECTRICAL" mountType="THROUGH_HOLE_PIN">
            <Location x="-3.81" y="-2.54"/>
            <Circle diameter="1.15"/>
            <Hole name="3" diameter="0.5" platingStatus="PLATED" plusTol="0.005"
minusTol="0.005" x="0" y="0"/>
        </Pin>
        <Pin number="4" type="THRU" electricalType="ELECTRICAL" mountType="THROUGH_HOLE_PIN">
            <Location x="3.81" y="-2.54"/>
            <Circle diameter="1.15"/>
            <Hole name="4" diameter="0.5" platingStatus="PLATED" plusTol="0.005"
minusTol="0.005" x="0" y="0"/>
        </Pin>
        <Pin number="5" type="THRU" electricalType="ELECTRICAL" mountType="THROUGH_HOLE_PIN">
            <Location x="3.81" y="0"/>
            <Circle diameter="1.15"/>
            <Hole name="5" diameter="0.5" platingStatus="PLATED" plusTol="0.005"
minusTol="0.005" x="0" y="0"/>
        </Pin>
        <Pin number="6" type="THRU" electricalType="ELECTRICAL" mountType="THROUGH_HOLE_PIN">
            <Location x="3.81" y="2.54"/>
            <Circle diameter="1.15"/>
            <Hole name="6" diameter="0.5" platingStatus="PLATED" plusTol="0.005"
minusTol="0.005" x="0" y="0"/>
        </Pin>
    </Package>
</PackageBlueprint>
```

Example: SOIC-8

The following XML fragment defines a package blueprint the SOIC-8 package.

```
<PackageBlueprint id="SOIC127P600-8N" name="SOIC-8" revisionDate="2012-07-17T21:32:52"
units="MILLIMETER">
    <Package name="SOIC-8" type="SOIC">
        <Outline>
            <LineDesc lineEnd="ROUND" lineWidth="0.001000"/>
            <Polygon>
                <PolyBegin x="-3.75" y="2.75"/>
                <PolyStepSegment x="-3.75" y="-2.75"/>
                <PolyStepSegment x="3.75" y="-2.75"/>
                <PolyStepSegment x="3.75" y="2.75"/>
                <PolyStepSegment x="-3.75" y="2.75"/>
            </Polygon>
        </Outline>
        <SilkScreen>
            <Outline>
                <LineDesc lineEnd="ROUND" lineWidth="0.2"/>
                <Polygon>
                    <PolyBegin x="-1.5" y="2.45"/>
                    <PolyStepSegment x="-1.5" y="-2.45"/>
                    <PolyStepSegment x="1.5" y="-2.45"/>
                    <PolyStepSegment x="1.5" y="2.45"/>
                    <PolyStepSegment x="-1.5" y="2.45"/>
                </Polygon>
            </Outline>
            <Marking markingUsage="PIN_ONE">
                <Location x="0" y="0"/>
                <Contour>
                    <Polygon>
                        <PolyBegin x="-1.5" y="2.45"/>
                        <PolyStepSegment x="-1.5" y="1.3"/>
                        <PolyStepSegment x="-0.35" y="2.45"/>
                        <PolyStepSegment x="-1.5" y="2.45"/>
                    </Polygon>
            </Contour>
            </Marking>
        </SilkScreen>
        <AssemblyDrawing>
            <Outline>
                <LineDesc lineEnd="ROUND" lineWidth="0.1"/>
                <Polygon>
                    <PolyBegin x="-1.95" y="2.45"/>
                    <PolyStepSegment x="-1.95" y="-2.45"/>
                    <PolyStepSegment x="1.95" y="-2.45"/>
                    <PolyStepSegment x="1.95" y="2.45"/>
                    <PolyStepSegment x="-1.95" y="2.45"/>
                </Polygon>
            </Outline>
            <Marking markingUsage="PIN_ONE">
                <Location x="-0.95" y="1.45"/>
                <Donut shape="ROUND" outerDiameter="1.0" innerDiameter="0.8"/>
            </Marking>
        </AssemblyDrawing>
        <Pin number="1" type="SURFACE" electricalType="ELECTRICAL"
mountType="SURFACE_MOUNT_PAD">
            <Location x="-2.65" y="1.905"/>
            <RectCenter width="1.65" height="0.6"/>
        </Pin>
        <Pin number="2" type="SURFACE" electricalType="ELECTRICAL"
mountType="SURFACE_MOUNT_PAD">
            <Location x="-2.65" y="0.635"/>
            <RectCenter width="1.65" height="0.6"/>
        </Pin>
        <Pin number="3" type="SURFACE" electricalType="ELECTRICAL"
mountType="SURFACE_MOUNT_PAD">
            <Location x="-2.65" y="-0.635"/>
            <RectCenter width="1.65" height="0.6"/>
        </Pin>
        <Pin number="4" type="SURFACE" electricalType="ELECTRICAL"
mountType="SURFACE_MOUNT_PAD">
            <Location x="-2.65" y="-1.905"/>
            <RectCenter width="1.65" height="0.6"/>
        </Pin>
        <Pin number="5" type="SURFACE" electricalType="ELECTRICAL"
mountType="SURFACE_MOUNT_PAD">
            <Location x="2.65" y="-1.905"/>
            <RectCenter width="1.65" height="0.6"/>
        </Pin>
        <Pin number="6" type="SURFACE" electricalType="ELECTRICAL"
mountType="SURFACE_MOUNT_PAD">
            <Location x="2.65" y="-0.635"/>
            <RectCenter width="1.65" height="0.6"/>
        </Pin>
        <Pin number="7" type="SURFACE" electricalType="ELECTRICAL"
mountType="SURFACE_MOUNT_PAD">
            <Location x="2.65" y="0.635"/>
            <RectCenter width="1.65" height="0.6"/>
        </Pin>
        <Pin number="8" type="SURFACE" electricalType="ELECTRICAL"
mountType="SURFACE_MOUNT_PAD">
            <Location x="2.65" y="1.905"/>
            <RectCenter width="1.65" height="0.6"/>
        </Pin>
    </Package>
</PackageBlueprint>
```

Chapter 8 Category Information for Component Organization

The ComponentCategoryDictionary element

ComponentCategoryDictionary

Contexts in which this element can be used:

As an element in the ComponentLibrary element

Content model:

Zero or more Category elements

Attributes:

dictionaryId

revisionDate

Defines a hierarchical structure for components. Components may exist in multiple categories, and applications may display the results merged. Tree nodes are described by nested Category elements that identify which components exist in the node.

Components not referenced by any Category element should appear uncategorized.

Attribute	Attribute Type	Description	Required
dictionaryId	notEmptyString	A unique identifier for the category tree.	Yes
revisionDate	xsd:dateTime	Date of this revision.	Yes

Component Category Dictionary Identifiers

This section is non-normative.

dictionaryId instances should be globally unique, although there is no registration process. In order to ensure uniqueness, dictionaryId instances should be written in reverse domain name notation. Examples of reverse domain name notation are:

```
com.domainname.analogcomponents
com.companyname
```

The Category element

Category

Contexts in which this element can be used:

As an element in the ComponentCategoryDictionary element or as an element in the Category element

Content model:

Zero or more LocalizedName elements followed by zero or more BlueprintRef elements followed by zero or more Category elements, followed by zero or more CategoryDictionaryRef elements

Attributes:

name

Defines an item (node) in the component category hierarchy. The name attribute provides a default name for the component category.

Attribute	Attribute Type	Description	Required
name	notEmptyString	A name for the family using US ASCII characters, as defined by IETF RFC 3986. Name must be unique within the context.	Yes

The following paragraph is non-normative.

Names containing characters not in the US ASCII character set can be specified by specifying a LocalizedName with an empty xml:lang attribute.

The LocalizedName element

LocalizedName

Contexts in which this element can be used:

As an element in the Category element

Content model:

Empty

Attributes:

xml:lang

name

Defines a translated name for the parent category.

Attribute	Attribute Type	Description	Required
xml:lang	Language identifier as defined by IETF BCP 47	The language for the localized name, or an empty string.	Yes
name	notEmptyString	The localized name for the category. Name must be unique within the context (sibling categories with the same name language identifier).	Yes

The BlueprintRef element

BlueprintRef

Contexts in which this element can be used:

As an element in the Category element

Content model:

Empty

Attributes:

blueprintRef

blueprintDateTime

Defines a reference to a component that exists at the particular node in the category.

Attribute	Attribute Type	Description	Required
blueprintRef	identifierRefType	The unique identifier for the component.	Yes
blueprintDateTime	xsd:dateTime	The dateTime of the referenced Blueprint. If omitted, matches the most recent Blueprint.	No

The CategoryDictionaryRef element

`CategoryDictionaryRef`

Contexts in which this element can be used:

As an element in the `Category` element

Content model:

Empty

Attributes:

`dictionaryId`

Defines a reference to a component that exists at the particular node in the category.

Attribute	Attribute Type	Description	Required
`dictionaryId`	`notEmptyString`	The unique identifier for the component category dictionary (see the `dictionaryId` attribute of the `CategoryDictionary` element).	Yes

Example

The following XML fragment defines a component category dictionary.

```
<ComponentCategoryDictionary dictionaryId="com.mycompany" revisionDate="2012-07-17T21:32:52">
    <Category name="Generic Manufacturer">
        <LocalizedName xml:lang="de" name="Generika-Hersteller"/>
        <Category name="Digital">
            <BlueprintRef blueprintRef="com.mycompany.comp.123"/>
            <BlueprintRef blueprintRef="com.mycompany.comp.456"/>
        </Category>
        <Category name="Analog">
            <BlueprintRef blueprintRef="com.mycompany.comp.789"/>
        </Category>
        <Category name="Educational">
            <LocalizedName xml:lang="de" name="Bildung"/>
            <!-- The component below is also in the Digital category -->
            <BlueprintRef blueprintRef="com.mycompany.comp.456"/>
        </Category>
        <CategoryDictionaryRef dictionaryId="com.companyinc.in-dev-components"/>
    </Category>
    <Category name="Components Inc">
        <BlueprintRef blueprintRef="com.mycompany.comp.ABC"/>
    </Category>
</ComponentCategoryDictionary>
```

Chapter 9 Components for Representing Orderable Items

The ComponentBlueprintDictionary element

`ComponentBlueprintDictionary`

Contexts in which this element can be used:

As an element in the `ComponentLibrary` element

Content model:

Zero or more `ComponentBlueprint` elements

Attributes:

None

Defines component blueprints as combinations of graphic symbols, simulation models and packages that when combined, represent an electronic component.

The ComponentBlueprint element

`ComponentBlueprint`

Contexts in which this element can be used:

As an element in the `ComponentBlueprintDictionary` element

Content model:

The following elements in any order

One `Connections` element

Zero or one `Sections` elements

One `SymbolConfigurations` element

Zero or one `SimulationModelConfigurations` elements

Zero or one `OrderablePackageConfigurations` elements

Zero or one `Properties` elements

Attributes:

`id`

`name`

`revisionDate`

The component blueprint is the main definition of a component. Component blueprint definitions in the component blueprint dictionary are not self-contained. Rather, component blueprints reference the other dictionaries in this or possibly other libraries. The primary role of the component blueprint is to tie these pieces together.

The component blueprint defines the connections on the component blueprint in the `Connections` and `Sections` sub-elements. Additionally, the component blueprint references items in other dictionaries (symbol, model and package) through the `SymbolConfigurations`, `SimulationModelConfigurations` and `PackageConfigurations` sub-elements, and defines the mapping between connections in the component and the pins, pads, ports, etc. in the mapped item.

The simplest `ComponentBlueprint` contains a list of connections (the `Connections` element) and an associated graphic symbol (the `SymbolConfigurations` element).

Attribute	Attribute Type	Description	Required
`id`	`identifierType`	Unique identifier for the component.	Yes
`name`	`notEmptyString`	Name of the component.	Yes
`revisionDate`	`xsd:dateTime`	Date of this revision.	Yes

The Connections element

`Connections`

Contexts in which this element can be used:

As an element in the `ComponentBlueprint` element

Content model:

One or more `Connection` elements

Attributes:

None

Defines the complete set of connections on the component. Each electrically independent connection to the component has a unique `Connection` element. For example, repeated ground connections only have one `Connection` element if the ground connections are internally connected. The connection list must include non-electrical connections exposed by a referenced simulation model in order to connect to the connection from a symbol.

The Connection element

`Connection`

Contexts in which this element can be used:

As an element in the `Connection` element

Content model:

Empty

Attributes:

`connName`

Defines the reference name for a connection on the component.

Attribute	Attribute Type	Description	Required
`connName`	`connectionNameType`	Identifier for the connection. This name is used as a reference within the component definition.	Yes

The Sections element

`Sections`

Contexts in which this element can be used:

As an element in the `ComponentBlueprint` element

Content model:

One or more `Section` elements

Attributes:

None

The `Sections` element groups connections and defines groups of connections that are functionally identical. Section order is not important, however, implementations should display connections in the same order as they occur in the `ComponentLibrary`.

Sections on the component group related pins. Sections define swappable units on a component. For example, a component with two AND gates would have two sections, one for each gate. The `Sections` element may be omitted for components with a single section.

The Section element

`Section`

Contexts in which this element can be used:

As an element in the `Sections` element

Content model:

One or more `ConnInstance` elements

Attributes:

`name`

`sectionSwapGroup`

Defines a grouping of connections by section.

Attribute	Attribute Type	Description	Required
name	notEmptyString	The section name is the name for mapping with graphic symbols, simulation models, and for display. Typically this is A, B, C, etc, or a functional description of the section. The name must be unique for a particular ComponentBlueprint. Required for components with multiple sections. May be omitted or empty value for single section components.	No
sectionSwapGroup	xsd:string	Sections with the name sectionSwapGroup value are functionally equivalent. An empty sectionSwapGroup does not match other empty sectionSwapGroup instances.	No

The ConnInstance element

ConnInstance

Contexts in which this element can be used:

As an element in the Section element

Content model:

Empty

Attributes:

connName

connSwapGroup

Identifies the connection as belonging to the ancestor section. Connections can belong to multiple sections (shared/common pins).

Attribute	Attribute Type	Description	Required
connName	connectionNameType	The name of the connection from the list in the Connections element.	Yes
connSwapGroup	xsd:string	Connections with the same connSwapGroup value are functionally equivalent within the same section. Identical connSwapGroup items identify functionally identical connections. A missing or empty connSwapGroup only matches identically named connections in other sections. If sectionSwapGroup is set for the containing Section, for the connections to be swappable all ConnectionInstance elements must have connSwapGroup specified and common names between Section elements, or identical connection names.	No

The SymbolConfigurations element

SymbolConfigurations

Contexts in which this element can be used:

As an element in the ComponentBlueprint element

Content model:

One or more SymbolConfiguration elements

Attributes:

None

Defines a set of graphical symbols for a ComponentBlueprint.

The SymbolConfiguration element

SymbolConfiguration

Contexts in which this element can be used:

As an element in the SymbolConfigurations element

Content model:

One or more Symbol elements

Attributes:

style

refdesPrefix

References and maps a particular graphical symbol for the ComponentBlueprint.

Attribute	Attribute Type	Description	Required
style	symbolStyleListType	The standards this graphical representation complies with.	Yes
refdesPrefix	prefixStringType	The character prefix for the RefDes, for example, IEC 81346 – Structuring principle and reference designators.	Yes

The Symbol element

Symbol

Contexts in which this element can be used:

As an element in the SymbolConfiguration element

Content model:

One or more PinMap elements

Attributes:

symbolRef

symbolDateTime

sectionName

Defines a reference to the the graphical symbol for a particular section on the component blueprint, and the mapping between connections and pins on the symbol.

Attribute	Attribute Type	Description	Required
symbolRef	identifierRefType	The unique identifier of the graphical symbol, referenced from the SymbolBlueprintDictionary.	Yes
symbolDateTime	xsd:dateTime	The dateTime of the referenced SymbolBlueprint. If omitted, matches the most recent SymbolBlueprint.	No
sectionName	xsd:string	Identifies which section this symbol represents. The value is the name of the section or missing/empty to indicate the graphic representation of all sections.	No

The PinMap element

PinMap

Contexts in which this element can be used:

As an element in the Symbol element

Content model:

Empty

Attributes:

connName

symbolPinId

displayName

Defines how connections on the component map to pin graphics on the symbol.

Attribute	Attribute Type	Description	Required
`connName`	`connectionNameType`	The name of the connection represented.	Yes
`symbolPinId`	`notEmptyString`	The ID of the pin in the `SymbolBlueprint`.	Yes
`displayName`	`notEmptyString`	The display name for the connection. This replaces the text on the `SymbolBlueprint`. If missing/empty, the `connName` attribute is used for the display name.	No

The following paragraph is non-normative.

Common practice shows inverted connections with lines above the characters. Overlines are drawn using the Unicode combining overline character (U+0305). Numeric superscript and subscript are similarly drawn using the Unicode superscript and subscript characters.

The SimulationModelConfigurations element

`SimulationModelConfigurations`

Contexts in which this element can be used:

As an element in the `ComponentBlueprint` element

Content model:

Zero or more `SimulationModelConfiguration` elements

Attributes:

None

Defines a set of simuation models for a `ComponentBlueprint`.

The SimulationModelConfiguration element

`SimulationModelConfiguration`

Contexts in which this element can be used:

As an element in the `SimulationModelConfigurations` element

Content model:

One or more `SimulationModel` elements

Attributes:

None

Defines a referemce to a particular simulation model for the component. The simulation model configuration is a complete set of simulation models for all sections on a component.

The SimulationModel element

`SimulationModel`

Contexts in which this element can be used:

As an element in the `SimulationModelConfiguration` element

Content model:

One or more `ModelPortMap` elements

Attributes:

`modelRef`

`modelDateTime`

`sectionName`

Defines the simulation model for a particular section on the component, or for all sections on the component. The distinction is determined by the value of the `sectionName` attribute.

Attribute	Attribute Type	Description	Required
modelRef	identifierRefType	The unique identifier of the simulation model, referenced from the SimulationModelBlueprintDictionary.	Yes
modelDateTime	xsd:dateTime	The dateTime of the referenced SimulationModelBlueprint. If omitted, matches the most recent SimulationModelBlueprint.	No
sectionName	xsd:string	Identifies which section this model represents. The value is the name of the section or missing/empty to indicate the model representation of all sections.	No

The ModelPortMap element

ModelPortMap

Contexts in which this element can be used:

As an element in the SimulationModel element

Content model:

Empty

Attributes:

connName

portName

Defines how connections on the component map to ports on the simulation model.

Attribute	Attribute Type	Description	Required
connName	connectionNameType	The name of the connection represented. The connection must belong to the section that this model represents.	Yes
portName	notEmptyString	The name of the port on the referenced simulation model being represented.	Yes

The OrderablePackageConfigurations element

OrderablePackageConfigurations

Contexts in which this element can be used:

As an element in the ComponentBlueprint element

Content model:

One or more OrderablePackageConfiguration elements

Attributes:

None

Defines a set of packages for a ComponentBlueprint.

The OrderablePackageConfiguration element

OrderablePackageConfiguration

Contexts in which this element can be used:

As an element in the OrderablePackageConfigurations element

Content model:

Zero or more PadMap elements

Attributes:

id

partNumber

manufacturer

manufacturerPackageRef

manufacturerPackageDateTime

Defines a particular orderable package for the component blueprint. Components are often available in multiple packages with nearly identical functionality, and the package configuration defines a particular orderable component.

Attribute	Attribute Type	Description	Required
`id`	`identifierType`	Unique identifier for the orderable package.	Yes
`partNumber`	`notEmptyString`	The part number to use to order the component with the particular package. The combination of the `partNumber` and `manufacturer` must be unique for the component blueprint.	Yes
`manufacturer`	`notEmptyString`	The manufacturer for the component. The combination of the `partNumber` and `manufacturer` must be unique for the component blueprint.	Yes
`manufacturerPackageRef`	`identifierRefType`	The unique identifier of the package, referenced from the `ManufacturerPackageDictionary`.	Yes
`manufacturerPackageDateTime`	`xsd:dateTime`	The `dateTime` of the referenced `ManufacturerPackage`. If omitted, matches the most recent `ManufacturerPackage`.	No

The PadMap element

`PadMap`

Contexts in which this element can be used:

As an element in the `OrderablePackageConfiguration` element

Content model:

Empty

Attributes:

`connName`

`padName`

Defines how connections on the component map to pads on the package.

Attribute	Attribute Type	Description	Required
connName	connectionNameType	The name of the connection represented. Connections can be repeated when they map to multiple distinct pads on the manufacturer package.	Yes
padName	notEmptyString	The name of the pad on the referenced manufacturer package being represented. Pad names must be unique in the mapping.	Yes

Example 1: resistor

The following XML fragment defines the component blueprint for a resistor.

```
<ComponentBlueprint id="com.example.comp.ABC123" name="ABC123">
    <Connections>
        <Connection connName="1"/>
        <Connection connName="2"/>
    </Connections>
    <Sections>
        <Section>
            <ConnInstance connName="1" connSwapGroup="1"/>
            <ConnInstance connName="2" connSwapGroup="1"/>
        </Section>
    </Sections>
    <SymbolConfigurations>
        <SymbolConfiguration style="IEEE_Std_315 IEC_60617" refdesPrefix="R">
            <Symbol symbolRef="com.example.symb.res"/>
                <PinMap connName="1" symbolPinId="1" displayName="1"/>
                <PinMap connName="2" symbolPinId="2" displayName="2"/>
            </Symbol>
        </SymbolConfiguration>
    </SymbolConfigurations>
    <SimulationModelConfigurations>
        <SimulationModelConfiguration>
            <SimulationModel modelRef="com.example.modl.res">
                <ModelPortMap connName="1" portName="1"/>
                <ModelPortMap connName="2" portName="2"/>
            </SimulationModel>
        </SimulationModelConfiguration>
    </SimulationModelConfigurations>
    <OrderablePackageConfigurations>
        <OrderablePackageConfiguration partNumber="ABC123-1" manufacturer="Components Inc" 
manufacturerPackageRef="com.example.mpkg.pkg1">
            <PadMap connName="1" padName="A"/>
            <PadMap connName="2" padName="B"/>
        </OrderablePackageConfiguration>
        <OrderablePackageConfiguration partNumber="ABC123-2" manufacturer="Components Inc" 
manufacturerPackageRef="com.example.mpkg.pkg2">
            <PadMap connName="1" padName="A"/>
            <PadMap connName="2" padName="B"/>
        </OrderablePackageConfiguration>
    </OrderablePackageConfigurations>
</ComponentBlueprint>
```

Example 2: electrolytic capacitor

The following XML fragment defines the component blueprint for an electrolytic capacitor.

```
<ComponentBlueprint id="com.example.comp.CBA123" name="CBA123">
    <Connections>
        <Connection connName="1"/>
        <Connection connName="2"/>
    </Connections>
    <SymbolConfigurations>
        <SymbolConfiguration style="IEEE_Std_315" refdesPrefix="C">
            <Symbol symbolRef="com.example.symb.cap">
                <PinMap connName="1" symbolPinId="1" displayName="1"/>
                <PinMap connName="2" symbolPinId="2" displayName="2"/>
            </Symbol>
        </SymbolConfiguration>
        <SymbolConfiguration style="IEC_60617" refdesPrefix="C">
            <Symbol symbolRef="com.example.symb.cap">
                <PinMap connName="1" symbolPinId="A" displayName="1"/>
                <PinMap connName="2" symbolPinId="B" displayName="2"/>
            </Symbol>
        </SymbolConfiguration>
    </SymbolConfigurations>
    <SimulationModelConfigurations>
        <SimulationModelConfiguration>
            <SimulationModel modelRef="com.example.modl.CBA123">
                <ModelPortMap connName="1" portName="1"/>
                <ModelPortMap connName="2" portName="2"/>
            </SimulationModel>
        </SimulationModelConfiguration>
    </SimulationModelConfigurations>
    <OrderablePackageConfigurations>
        <OrderablePackageConfiguration partNumber="CBA123-1" manufacturer="Components Inc"
manufacturerPackageRef="com.example.mpkg.CBA123-1">
            <PadMap connName="1" padName="A"/>
            <PadMap connName="2" padName="B"/>
        </OrderablePackageConfiguration>
    </OrderablePackageConfigurations>
</ComponentBlueprint>
```

Example 3: electrolytic capacitor #2

The following XML fragment defines the component blueprint for an electrolytic capacitor. This definition is equivalent to Example 2.

```
<ComponentBlueprint id="com.example.comp.cap" name="Electrolytic Capacitor">
    <Connections>
        <Connection connName="1"/>
        <Connection connName="2"/>
    </Connections>
    <Sections>
        <Section name="0">
            <ConnInstance connName="1" connGroup="1"/>
            <ConnInstance connName="2" connGroup="2"/>
        </Section>
    </Sections>
    <SymbolConfigurations>
        <SymbolConfiguration style="IEEE_Std_315 IEC_60617" refdesPrefix="C">
            <Symbol symbolRef="com.example.symb.generic-cap">
                <PinMap connName="1" symbolPinId="1" displayName="1"/>
                <PinMap connName="2" symbolPinId="2" displayName="2"/>
            </Symbol>
        </SymbolConfiguration>
    </SymbolConfigurations>
    <SimulationModelConfigurations>
        <SimulationModelConfiguration>
            <SimulationModel modelRef="com.example.modl.generic-cap">
                <ModelPortMap connName="1" portName="1"/>
                <ModelPortMap connName="2" portName="2"/>
            </SimulationModel>
        </SimulationModelConfiguration>
    </SimulationModelConfigurations>
    <OrderablePackageConfigurations>
        <OrderablePackageConfiguration partNumber="CBA123-1" manufacturer="Components Inc"
manufacturerPackageRef="com.example.mpkg.CBA123-1">
            <PadMap connName="1" padName="A"/>
            <PadMap connName="2" padName="B"/>
        </OrderablePackageConfiguration>
    </OrderablePackageConfigurations>
</ComponentBlueprint>
```

Example 4: quad 2-input positive NAND gates

The following XML fragment defines the component blueprint for the 7400N quad 2-input positive NAND gates. On the second symbol configuration, some pins are not visible and require hidden connections.

```
<ComponentBlueprint id="com.example.comp.7400N" name="7400N">
    <Connections>
        <Connection connName="1A"/>
        <Connection connName="1B"/>
        <Connection connName="1Y"/>
        <Connection connName="2A"/>
        <Connection connName="2B"/>
        <Connection connName="2Y"/>
        <Connection connName="3A"/>
        <Connection connName="3B"/>
        <Connection connName="3Y"/>
        <Connection connName="4A"/>
        <Connection connName="4B"/>
        <Connection connName="4Y"/>
        <Connection connName="VCC"/>
        <Connection connName="GND"/>
    </Connections>
    <Sections>
        <Section name="1" sectionSwapGroup="1">
            <ConnInstance connName="1A" connSwapGroup="1"/>
            <ConnInstance connName="1B" connSwapGroup="1"/>
            <ConnInstance connName="1Y" connSwapGroup="2"/>
            <ConnInstance connName="VCC"/>
            <ConnInstance connName="GND"/>
        </Section>
        <Section name="2" sectionSwapGroup="1">
            <ConnInstance connName="2A" connSwapGroup="1"/>
            <ConnInstance connName="2B" connSwapGroup="1"/>
            <ConnInstance connName="2Y" connSwapGroup="2"/>
            <ConnInstance connName="VCC"/>
            <ConnInstance connName="GND"/>
        </Section>
        <Section name="3" sectionSwapGroup="1">
            <ConnInstance connName="3A" connSwapGroup="1"/>
            <ConnInstance connName="3B" connSwapGroup="1"/>
            <ConnInstance connName="3Y" connSwapGroup="2"/>
            <ConnInstance connName="VCC"/>
            <ConnInstance connName="GND"/>
        </Section>
        <Section name="4" sectionSwapGroup="1">
            <ConnInstance connName="4A" connSwapGroup="1"/>
            <ConnInstance connName="4B" connSwapGroup="1"/>
                <ConnInstance connName="4Y" connSwapGroup="2"/>
            <ConnInstance connName="VCC"/>
            <ConnInstance connName="GND"/>
        </Section>
    </Sections>
    <SymbolConfigurations>
        <SymbolConfiguration style="IEEE_Std_315 IEC_60617" refdesPrefix="U">
            <Symbol symbolRef="com.example.symb.nand-quad">
                <PinMap connName="1A" symbolPinId="1" displayName="A"/>
                <PinMap connName="1B" symbolPinId="2" displayName="B"/>
                <PinMap connName="1Y" symbolPinId="3" displayName="Y"/>
                <PinMap connName="2A" symbolPinId="4" displayName="A"/>
                <PinMap connName="2B" symbolPinId="5" displayName="B"/>
                <PinMap connName="2Y" symbolPinId="6" displayName="Y"/>
                <PinMap connName="3A" symbolPinId="7" displayName="A"/>
                <PinMap connName="3B" symbolPinId="8" displayName="B"/>
                <PinMap connName="3Y" symbolPinId="9" displayName="Y"/>
                <PinMap connName="4A" symbolPinId="10" displayName="A"/>
                <PinMap connName="4B" symbolPinId="11" displayName="B"/>
                <PinMap connName="4Y" symbolPinId="12" displayName="Y"/>
                <PinMap connName="VCC" symbolPinId="13" displayName="VCC"/>
                <PinMap connName="GND" symbolPinId="14" displayName="GND"/>
            </Symbol>
        </SymbolConfiguration>
        <SymbolConfiguration style="IEEE_Std_315 IEC_60617" refdesPrefix="U">
            <Symbol symbolRef="com.example.symb.nand" sectionName="1">
                <PinMap connName="1A" symbolPinId="1" displayName="1A"/>
                <PinMap connName="1B" symbolPinId="2" displayName="1B"/>
                <PinMap connName="1Y" symbolPinId="3" displayName="1C"/>
            </Symbol>
            <Symbol symbolRef="com.example.symb.nand" sectionName="2">
                <PinMap connName="2A" symbolPinId="1" displayName="2A"/>
                <PinMap connName="2B" symbolPinId="2" displayName="2B"/>
                <PinMap connName="2Y" symbolPinId="3" displayName="2C"/>
            </Symbol>
            <Symbol symbolRef="com.example.symb.nand" sectionName="3">
                <PinMap connName="3A" symbolPinId="1" displayName="3A"/>
                <PinMap connName="3B" symbolPinId="2" displayName="3B"/>
                <PinMap connName="3Y" symbolPinId="3" displayName="3C"/>
                </Symbol>
            <Symbol symbolRef="com.example.symb.nand" sectionName="4">
                <PinMap connName="4A" symbolPinId="1" displayName="4A"/>
                <PinMap connName="4B" symbolPinId="2" displayName="4B"/>
                <PinMap connName="4Y" symbolPinId="3" displayName="4C"/>
            </Symbol>
        </SymbolConfiguration>
    </SymbolConfigurations>
    <SimulationModelConfigurations>
        <SimulationModelConfiguration>
            <SimulationModel modelRef="com.example.modl.nand-quad">
                <ModelPortMap connName="1A" portName="1"/>
                <ModelPortMap connName="1B" portName="2"/>
                <ModelPortMap connName="1Y" portName="3"/>
                <ModelPortMap connName="2A" portName="4"/>
                <ModelPortMap connName="2B" portName="5"/>
            <ModelPortMap connName="2Y" portName="6"/>
            <ModelPortMap connName="3A" portName="7"/>
                <ModelPortMap connName="3B" portName="8"/>
                <ModelPortMap connName="3Y" portName="9"/>
                <ModelPortMap connName="4A" portName="10"/>
                <ModelPortMap connName="4B" portName="11"/>
                     <ModelPortMap connName="4Y" portName="12"/>
                <ModelPortMap connName="VCC" portName="13"/>
                <ModelPortMap connName="GND" portName="14"/>
            </SimulationModel>
        </SimulationModelConfiguration>
    </SimulationModelConfigurations>
    <OrderablePackageConfigurations>
        <OrderablePackageConfiguration partNumber="SN5400N" manufacturer="Comp Manuf Inc"
manufacturerPackageRef="com.example.ordpkg.SN4500N">
            <PadMap connName="1A" padName="1"/>
            <PadMap connName="1B" padName="2"/>
            <PadMap connName="1Y" padName="3"/>
            <PadMap connName="2A" padName="4"/>
            <PadMap connName="2B" padName="5"/>
            <PadMap connName="2Y" padName="6"/>
            <PadMap connName="GND" padName="7"/>
            <PadMap connName="3A" padName="8"/>
            <PadMap connName="3B" padName="9"/>
            <PadMap connName="3Y" padName="10"/>
                <PadMap connName="4A" padName="11"/>
            <PadMap connName="4B" padName="12"/>
            <PadMap connName="4Y" padName="13"/>
            <PadMap connName="VCC" padName="14"/>
        </OrderablePackageConfiguration>
        <OrderablePackageConfiguration partNumber="SNJ5400W" manufacturer="Comp Manuf Inc"
manufacturerPackageRef="com.example.ordpkg.SNJ4500W">
            <PadMap connName="1A" padName="1"/>
            <PadMap connName="1B" padName="2"/>
            <PadMap connName="1Y" padName="3"/>
            <PadMap connName="GND" padName="4"/>
            <PadMap connName="2A" padName="5"/>
            <PadMap connName="2B" padName="6"/>
            <PadMap connName="2Y" padName="7"/>
            <PadMap connName="3A" padName="8"/>
            <PadMap connName="3B" padName="9"/>
            <PadMap connName="3Y" padName="10"/>
            <PadMap connName="VCC" padName="11"/>
            <PadMap connName="4A" padName="12"/>
            <PadMap connName="4B" padName="13"/>
            <PadMap connName="4Y" padName="14"/>
        </OrderablePackageConfiguration>
        <OrderablePackageConfiguration partNumber="FJH131" manufacturer="Part Manuf Inc"
manufacturerPackageRef="com.example.ordpkg.FJH131">
            <PadMap connName="1A" padName="1"/>
            <PadMap connName="1B" padName="2"/>
            <PadMap connName="1Y" padName="3"/>

                    <PadMap connName="2A" padName="4"/>

            <PadMap connName="2B" padName="5"/>
            <PadMap connName="2Y" padName="6"/>
            <PadMap connName="GND" padName="7"/>
            <PadMap connName="3A" padName="8"/>
            <PadMap connName="3B" padName="9"/>
            <PadMap connName="3Y" padName="10"/>
            <PadMap connName="4A" padName="11"/>
            <PadMap connName="4B" padName="12"/>
            <PadMap connName="4Y" padName="13"/>
            <PadMap connName="VCC" padName="14"/>
        </OrderablePackageConfiguration>
    </OrderablePackageConfigurations>
</ComponentBlueprint>
```

Chapter 10 Status Information for Component Obsolescence

The StatusDictionary element

StatusDictionary

Contexts in which this element can be used:

As an element in the ComponentLibrary element

Content model:

Zero or more Status elements

Attributes:

None

Defines obsolescence (availability) information for components.

The Status element

Status

Contexts in which this element can be used:

As an element in the StatusDictionary element

Content model:

Zero or more Replacement elements

Attributes:

ref

status

comment

The Status element describes the default availability/obsolescence of a ComponentBlueprint or a specific availability/obsolescence of particular PackageConfiguration. Additionally, the Status may define replacement ComponentBlueprint/PackageConfiguration items. The Status element references the ComponentBlueprint/OrderablePackageConfiguration by ID.

ComponentBlueprint and OrderablePackageConfiguration instances without status information are defined to be available without restriction.

Attribute	Attribute Type	Description	Required
ref	identifierRefType	Reference to the unique identifier for the ComponentBlueprint or PackageConfiguration.	Yes
status	statusType	Defines the default status for all part numbers in the component blueprint.	Yes
comment	xsd:string	Implementations may use this field to indicate the source of the status information.	No

Defines status information for a particular part number from a manufacturer. This element describes the manufacturer's definition of status/availability, as opposed to the availability from a particular supplier/distributor.

The Replacement element

Replacement

Contexts in which this element can be used:

As an element in the Status element

Content model:

Empty

Attributes:

ref

Defines a replacement for an obsolete part.

Attribute	Attribute Type	Description	Required
ref	identifierRefType	Reference to the unique identifier for the ComponentBlueprint or PackageConfiguration of the replacement.	Yes

Chapter 11 Order Information for Connections to Suppliers/Distributors

The OrderingInfoDictionary element

OrderingInfoDictionary

Contexts in which this element can be used:

As an element in the ComponentLibrary element

Content model:

Zero or more OrderingInfo elements

Attributes:

None

Defines information about orderable components, for example from 3rd party distributors. This information can be included in the OECL, but often exists in 3rd party repositories. The order information is linked to the orderable component by a unique identifier.

The OrderingInfo element

OrderingInfo

Contexts in which this element can be used:

As an element in the OrderingInfoDictionary element

Content model:

Zero or more Price elements

Attributes:

supplierSku

supplierName

supplierUrl

availability

quantityAvailable

orderablePackageRef

Defines purchase information about a particular component. The OrderInfo element is associated with the particular component through the orderablePackageRef attribute, allowing for multiple OrderInfo elements for the same component.

Attribute	Attribute Type	Description	Required
`supplierSku`	`xsd:string`	The supplier's SKU for the component.	Yes
`supplierName`	`xsd:string`	The name of the supplier	Yes
`supplierUrl`	`xsd:anyURL`	Supplier specific URL for the component.	No
`availability`	`availabilityType`	Availability of the component from the supplier. Omitted attribute is equivalent to `unknown`	No
`quantityAvailable`	`quantityAvailableType`	The quantity that the supplier has available. Omitted attribute is equivalent to `unknown`.	No
`orderablePackageRef`	`identifierRefType`	Reference to the unique identifier for the `OrderablePackageConfiguration`.	Yes

The Price Element

`Price`

Contexts in which this element can be used:

As an element in the `OrderingInfo` element

Content model:

Empty

Attributes:

`minimumQuantity`

`price`

`currency`

Defines pricing for ordering. Prices are grouped into ranges.

Attribute	Attribute Type	Description	Required
minimumQuantity	positiveDecimalType	The minimum number of items for this unit price.	Yes
unitPrice	xsd:decimal	The price per unit.	Yes
currency	currencyType	The currency using the ISO 4217 alphabetic code (three-character),	Yes

Example

The following example defines ordering information from two separate suppliers/distributors or the same component (manufacturer part number).

```
<OrderingInfo supplierSku="ABC-123" supplierName="Part Distributor Inc"
supplierUrl="http://www.example.com/ABC-123" availability="inStock" quantityAvailable="1000"
orderablePackageRef="com.example.ordpkg.ABC-123">
    <Price minimumQuantity="1" unitPrice="2.00" currency="USD"/>
    <Price minimumQuantity="100" unitPrice="1.50" currency="USD"/>
    <Price minimumQuantity="1000" unitPrice="1.25" currency="USD"/>
</OrderingInfo>
<OrderingInfo supplierSku="123-456-789" supplierName="Local Parts Inc"
supplierUrl="http://www.example.org/123-456-789" availability="unknown"
quantityAvailable="unknown" orderablePackageRef="org.example.ordpkg.123-456-789">
    <Price minimumQuantity="1" unitPrice="2.10" currency="USD"/>
    <Price minimumQuantity="100" unitPrice="1.55" currency="USD"/>
    <Price minimumQuantity="1000" unitPrice="1.20" currency="USD"/>
</OrderingInfo>
```

Chapter 12 Standard Property Keys

The OECL Format allows extensions by defining standard property keys. Standard property keys allow applications to assign meaning to the property and perform application-specific processing on the property.

Consumers of OECL documents should use these standard property keys when processing the document.

Short Description

property element type	`TextProperty`
key	`short_description`

Short description of a `ComponentBlueprint` with 160 or fewer characters.

Example

```
<TextProperty key="short_description" name="Description" value="The value describes the
component blueprint in fewer than 160 characters."/>
```

Description

property element type	`TextProperty`
key	`description`

Description of a `ComponentBlueprint`.

Example

```
<TextProperty key="description" name="Description" value="The value describes the component
blueprint. The description text could be very long."/>
```

Datasheet

property element type	`URLProperty`
key	`datasheet`

Link to the datasheet for the `ComponentBlueprint`. The link is directly to the datasheet, not a landing page with further links.

Example

```
<URLProperty key="datasheet" name="Datasheet" value="http://www.example.com/datasheet.pdf"/>
```

Chapter 13 References

IEEE Std 315

IEEE. Graphical Symbols for Electrical and Electronics Diagrams (Including Reference Designation Letters). 1975.

IETF BCP 47

IETF (Internet Engineering Task Force). *BCP* 47, *consisting of RFC 4646: Tags for Identifying Languages, and RFC 4647: Matching of Language Tags*, A. Phillips, M. Davis. 2006. (See http://tools.ietf.org/html/bcp47.)

IETF RFC 3986

IETF (Internet Engineering Task Force). *RFC 3986: Uniform Resource Identifier (URI): Generic Syntax*. Berners-Lee, T., Fielding, R., and Masinter, L, editors. Internet Engineering Task Force. 2005. (See http://www.ietf.org/rfc/rfc3986.txt.)

IPC-2581A

IPC. *Generic Requirements for Printed Board Assembly Products Manufacturing Description Date and Transfer Methodology*. 2012. (See http://www.ipc-2581.com for the latest version.)

IPC-7351

IPC. IPC-7351 *Generic Requirements for Surface Mount Design and Land Pattern Standard*. 2005. (See http://www.ipc.org/TOC/IPC-7351.pdf.)

ISO 4217

ISO (International Organization for Standardization). *ISO 4217:2008. Codes for the representation of currencies and funds*, as, from time to time, amended, replaced by a new edition or expanded by the addition of new parts. [Geneva]: International Organization for Standardization. (See http://www.iso.org for the latest version.)

IEC 60617

IEC (International Electrotechnical Comission). *IEC 60617 – Graphical Symbols for Diagrams*.

IEC 81346

ISO (International Organization for Standardization). IEC 81346-1:2009. Industrial systems, installations and equipment and industrial products - Structuring principles and reference designations – Part 1: Basic rules. (See http://www.iso.org for the latest version.)

HTML5

W3C (World Wide Web Consortium). HTML5 A vocabulary and associated APIs for HTML and XHTML. W3C Working Draft 25 October 2012. (See http://www.w3.org/TR/2012/WD-html5-20121025/)

SVG

W3C (World Wide Web Consortium). Scalable Vector Grapics (SVG) 1.1 (Second Edition). W3C Recommendation 16 August 2011. (See http://www.w3.org/TR/2011/REC-SVG11-20110816/)