EcoreDoc User Guide

EC Modeling & Simulation

Version 2024-11-26 15:12 UTC

Table of Contents

Generates AsciiDoctor files to document Ecore metamodels, similar to JavaDoc. AsciiDoctor can be rendered as HTML, PDF, or Eclipse Help. EcoreDoc can be used as Maven Plugin, standalone command-line tool, Java API, or Eclipse Plug-in.

Chapter 1. Overview

EcoreDoc's *Java API* works on a list of `EClassifier`s. *Maven Plugin*, *Standalone Command-Line Tool* and *Eclipse Plug-in* take all `EClassifier`s from one or more `*.ecore` or `*.xcore` files.

EcoreDoc creates one output document containing all passed `EClassifier`s. They are grouped by containing `EPackage`. The output document contains documentation, all properties of supported elements, and cross-references to all usages of each element.

EcoreDoc currently supports the following elements:

- `EPackage`
- `EDataType`
- `EEnum`
- `EEnumLiteral`
- `EClass`
- `EAttribute`
- `EReference`
- `EOperation`
- `EParameter`

EcoreDoc is highly configurable via the *Generator Configuration*.

The homepage and repository of EcoreDoc can be found at https://github.com/altran-mde/ecore_doc. Please use the issue tracker at this site for any feature requests or bugs.

Chapter 2. Java API

The Java API is available as Maven artifact and OSGi bundle `com.altran.general.emf.ecoredoc:com.altran.general.emf.ecoredoc.generator`.
The *Generator Configuration* is contained in Maven artifact and OSGi bundle `com.altran.general.emf.ecoredoc:com.altran.general.emf.ecoredoc.generator.config`.

The main interface is `com.altran.general.emf.ecoredoc.generator.EcoreDocGenerator`. The constructor takes the list of `EClassifier`s to generate documentation for.

The `getConfig()` method returns a fully initialized `com.altran.general.emf.ecoredoc.generator.config.EcoreDocGeneratorConfig` that can be changed to adjust the *Generator Configuration*.

The `generate()` method returns a `CharSequence` containing the complete AsciiDoctor output document.

Chapter 3. Maven Plugin

The Maven Plugin is available as Maven artifact `com.altran.general.emf.ecoredoc:ecoredoc-maven-plugin`.

It supports the following configuration settings:

inputFiles (required)

The list of Ecore metamodel files to create documentation for.

outputFile (required)

The output file to write the generated AsciiDoctor document to.
By convention, the file extension is `.adoc`.

> 🔥 If the file exists, it will be overwritten and a warning is emitted.

resolve (default: false)

Whether EcoreDoc should explicitly try to resolve all references in the *inputFiles*. Might be necessary for highly interconnected metamodels.

config (default: unchanged default config)

Customized *Generator Configuration*.

The *config* contents stricly follow the structure and naming relative to Class EcoreDocGeneratorConfig, easiest explained with an example.

Assume the *inputFiles* contain two `EPackage`s, namely `EPackage1` and `EPackage2`.
`EPackage1` contains, among others, two `EClass`es, named `MyEClass` and `Class3`. The latter one contains, among others, the `EAttribute` named `specialNumber`.
`EPackage1` also contains an `EEnum` named `Enum1`.

pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <artifactId>my-artifact-id</artifactId>

  <dependencies>
    <dependency>
      <groupId>com.altran.general.emf.ecoredoc</groupId>
      <artifactId>ecoredoc-maven-plugin</artifactId>
    </dependency>
  </dependencies>
```

```
<build>
  <plugins>
    <plugin>
      <groupId>com.altran.general.emf.ecoredoc</groupId>
      <artifactId>ecoredoc-maven-plugin</artifactId>

      <!-- make sure EcoreDoc is actually executed -->
      <executions>
        <execution>
          <phase>test</phase>
          <goals>
            <goal>ecoredoc</goal>
          </goals>
        </execution>
      </executions>

      <configuration>
        <resolve>true</resolve>

        <config>
          <renderDefaults>false</renderDefaults>
          <documentTitle>This is the title of my document</documentTitle>
          <ePackages>
            <ePackage>
              <targetEPackage>EPackage1</targetEPackage>
              <eClasses>
                <eClass>
                  <targetEClass>MyEClass</targetEClass>
                  <repeatInherited>false</repeatInherited>
                </eClass>
                <eClass>
                  <targetEClass>Class3</targetEClass>
                  <eAttributes>
                    <eAttribute>
                      <targetEAttribute>specialNumber</targetEAttribute>
                      <render>false</render>
                    </eAttribute>
                  </eAttributes>
                </eClass>
              </eClasses>
              <eEnums>
                <eEnum>
                  <targetEEnum>Enum1</targetEEnum>
                  <renderDefaults>true</renderDefaults>
                </eEnum>
              </eEnums>
            </ePackage>
            <ePackage>
              <targetEPackage>EPackage2</targetEPackage>
              <renderDefaults>true</renderDefaults>
            </ePackage>
            </ePackages>
          </config>
          <inputFiles>
            <inputFile>EPackage1.ecore</inputFile>
            <inputFile>EPackage2.ecore</inputFile>
          </inputFiles>
          <outputFile>output.adoc</outputFile>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

This example sets the following configuration:

- renderDefaults for all contents: `false`
- documentTitle: `This is the title of my document`
- repeatInherited for *MyEClass*: `false`
- render for *specialNumber*: `false`
- renderDefaults for *Enum1*: `true`
- renderDefaults for *EPackage2*: `true`

Chapter 4. Standalone Command-Line Tool

The standalone command-line tool is available as Maven artifact `com.altran.general.emf.ecoredoc:com.altran.general.emf.ecoredoc.standalone`.

Use the following command to invoke. Please replace `${ecoredoc-version}` with your version of EcoreDoc:

```
java -jar com.altran.general.emf.ecoredoc.standalone-${ecoredoc-version}-jar-with
-dependencies.jar <options>
```

If invoked without options, it will print the following help:

```
Generates reference documentation for ecore models.

The output is inspired by JavaDoc and formatted in AsciiDoctor format.
AsciiDoctor can easily be rendered to HTML, PDF, or Eclipse help.


Usage:

EcoreDocGenerator [parameters] [List of ecore files to generate]

If unspecified, the output file name will be "<firstEcoreFile.ecore>.adoc"


Parameters:

  -r,
  --resolve: Resolve external references

  -o <outputFile>,
  --output <outputFile>: Specify output file name.

  --documentTitle <title>: Set title of output document

  --positionEDataTypes <pos>: Set rendering position of all EDataTypes within EPackage

  --positionEEnums <pos>:     Set rendering position of all EEnums within EPackage

  --positionEClasses <pos>:   Set rendering position of all EClasses within EPackage

  [+|-]defaults:   [Enable|disable] rendering of default values

  [+|-]bounds:     [Enable|disable] rendering of multiplicity bounds
                                    (overwrites defaults parameter)
  [+|-]inherited:  [Enable|disable] repetition of inherited features

  [+|-]useCases:   [Enable|disable] rendering of use cases
                                    (references to other usages of this element)


  [+|-]subTypes:   [Enable|disable] rendering of sub-types

  [+|-]superTypes: [Enable|disable] rendering of super-types

  [+|-]diagrams:   [Enable|disable] rendering of diagrams

Examples:

  EcoreDocGenerator my.ecore
  Generates the documentation of my.ecore into my.ecore.adoc


  EcoreDocGenerator some/path/to/my.ecore other.ecore
  Generates the documentation of some/path/to/my.ecore and other.ecore
  into some/path/to/my.ecore.adoc


  EcoreDocGenerator -r my.ecore
  Tries to resolve all external references in my.ecore and
  generates the documentation of my.ecore and referenced models into my.ecore.adoc


  EcoreDocGenerator -defaults +bounds my.ecore
  Generates the documentation of my.ecore and referenced models into my.ecore.adoc
  without rendering default values, but still rendering multiplicity bounds


  EcoreDocGenerator --positionEClasses 1 --positionEEnums 2 --positionEDataTypes 3
my.ecore
  Generates the documentation of my.ecore and referenced models into my.ecore.adoc
  with all EClasses first, then all EEnums, and finally all EDataTypes


  EcoreDocGenerator -o output.adoc my.ecore other.ecore
  Generates the documentation of my.ecore and other.ecore into output.adoc
```

Chapter 5. Eclipse Plug-in

The Eclipse Plug-in is available as Feature `com.altran.general.emf.ecoredoc.ui.feature`.

It provides a context menu entry for one or more `*.ecore` / `*.xcore` files in the following views:

- Project Explorer
- Package Explorer
- Model Explorer

The command creates one output file next to the first selected input file, named `<firstInputFile.ecore>.adoc`. The output file contains the documentation of all selected metamodels.

Chapter 6. Eclipse Generation Factories Plug-in

The EGF (Eclipse Generation Factories) Plug-in is available as Feature `com.altran.general.emf.ecoredoc.egf.feature`.

EGF is a software factory tool with the purpose to generate software artifacts, such as code or application, in order to automate software development. The EcoreDoc EGF plug-in provides a *factory component* to generate EcoreDoc for one `*.ecore` file. This EcoreDoc *factory component* is also integrated in the *generation chain* for `*.ecore` files. More information on using *generation chains* can be found in the EGF documentation

Both the factory component as the generation chain support the following configuration settings:

domain (required)

The Ecore metamodel file to create documentation for.

projectName (required)

The Eclipse project name to use for generation. Combined with outputFolder this denotes the location to write the generated AsciiDoctor document to.

outputFolder (required)

The output folder to write the generated AsciiDoctor document to.
By convention, the generated file is `<projectName>/<outputFolder>/<domain_file_name>.adoc`.

> 🔥 If the file exists, it will be overwritten and a warning is emitted.

resolve (default: false)

Whether EcoreDoc should explicitly try to resolve all references in the domain. Might be necessary for highly interconnected metamodels.

Chapter 7. EcoreDoc Metamodel Annotation

Any of the *Generator Configuration* options can be used as Ecore Annotation. These options will be used by default; any external options take precedence over annotation options.

EcoreDoc will throw an `IllegalArgumentException` if an EcoreDoc annotation contains an illegal value.

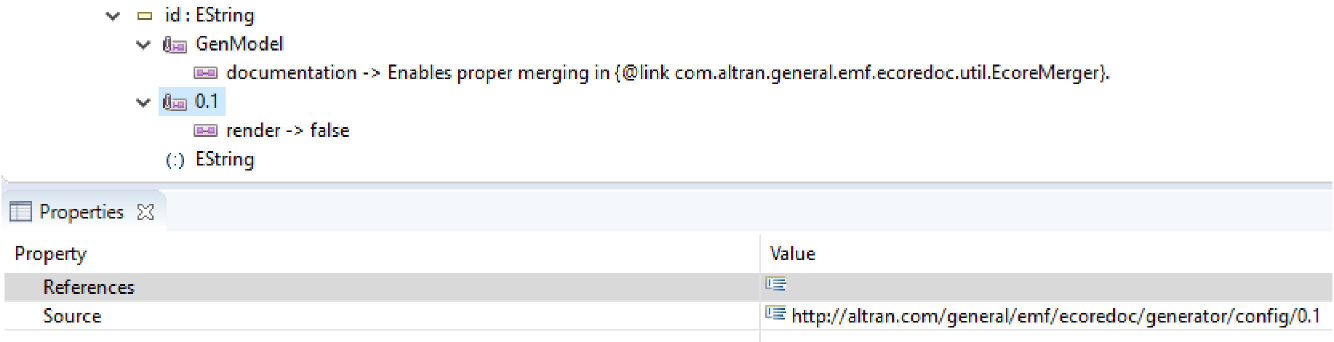
7.1. Ecore Annotation

Create an `EAnnotation` on the annotated element with source

```
http://altran.com/general/emf/ecoredoc/generator/config/0.1
```

Within this annotation, create one key/value pair for each option.

> The source identifier might change in the future! However, it should be possible to maintain backwards compatibility.



```
id : EString
  GenModel
    documentation -> Enables proper merging in {@link com.altran.general.emf.ecoredoc.util.EcoreMerger}.
  0.1
    render -> false
  EString
```

Property	Value
References	
Source	http://altran.com/general/emf/ecoredoc/generator/config/0.1

Example Ecore annotation on EAttribute **id** *(sets render for* **id** *to* `false`*)*

7.2. Xcore Annotation

First, register the annotation.

EcoreDoc annotation registration

```
annotation "http://altran.com/general/emf/ecoredoc/generator/config/0.1" as EcoreDoc
```

Afterwards, we can use the annotation as usual.

Example Xcore annotation on EAttribute **name** *(sets render for* **name** *to* `false`*)*

```
@EcoreDoc(
  render="false"
)
String name
```

Chapter 8. EOperation Overrides

EcoreDoc understands all possible kinds of inheritance and overrides and marks them accordingly.

We use the following example throughout the section.

```
class SomeClass {
  String myName
  contains SomeClass[0..*] others
}

class Class3 extends SomeClass {}

interface IFace1 {
  op void doIt()

  op void doIt(int i)
}

interface IFace2 {
  op void doIt()
}

abstract class AImplementer extends IFace1, IFace2 {
  op void doIt() {
    println("Hello, World!")
  }
}

interface IFace3 extends IFace1, IFace2 {}

class Implementer extends AImplementer, IFace3 {}

interface If1 {
  op Class3[1..8] getChildren()
}

interface If2 {
  op SomeClass getSome()
  op void setSome(SomeClass[1] someClass)
}

class Cls3 extends If1, If2 {
  contains Class3[] children

  refers SomeClass[1] some
}

class Cl4 {
  refers If1 iface
}

class Cl4b extends Cl4 {
  op Cls3 getIface() {
    super.iface as Cls3
  }

  op void setIface(Cls3 iface) {
    super.iface = iface
  }
}
```

Show inherited Features

If repeatInherited is enabled, we repeat all inherited features (i.e. *EAttribute*s, containing *EReference*s, cross-referencing *EReference*s) from all super-types.

We link them to their declaration with symbol ▯ .

We omit inherited features if they are overridden by an *EOperation* (see below).

In the example, we repeat `SomeClass.myName` and `SomeClass.others` in `Class3`.

Show inherited EOperations

If repeatInherited is enabled, we repeat all inherited *EOperation*s from all super-types.

If several super-types declare the same *EOperation* (compared by signature), we repeat this *EOperation* only once and link to all the declarations with symbol ▯ . If one of the declarations defines a body, we repeat that body.

We omit inherited *EOperation*s if they are overridden by a Feature (see below).

In the example, we repeat both `Iface1.doIt()` and `Iface2.doIt()` once in `Iface3`, linking to both super-types. We also repeat `Iface1.doIt(i)` in `Iface3`.

Show overridden EOperations

If an *EOperation* defines a body and one or more super-types declare the same *EOperation* (compared by signature), we link to all the super-type declarations with symbol ▯ .

In the example, we mark `AImplementer.doIt()` as overriding `Iface1.doIt()` and `Iface2.doIt()`.

Show overriding EOperations

If an *EOperation* is declared in one or more sub-types and they define a body, we link to all sub-types declarations with symbol ▯ .

In the example, we mark both `Iface1.doIt()` and `Iface2.doIt()` as being overridden by `AImplementer.doIt()`.

Show Features overriding EOperation

If the generated code for a feature effectively overrides one or more inherited *EOperation*s, we link from the feature to all overridden *EOperation*s with symbol ▯ .

In this case, we omit the inherited and overridden *EOperation*s.

We also link to all features of all sub-types overriding an *EOperation* with symbol □ .

In the example, we mark `Cls3.children` as overriding `If1.getChildren()`, and `Cls3.some` as overriding both `If2.getSome()` and `If2.setSome()`.
Accordingly, we mark `If1.getChildren()` as being overridden by `Cls3.children`, and both `If2.getSome()` and `If2.setSome()` as being overridden by `Cls3.some`.
We also omit all the *EOperation*s from `Cls3`, as they are effectively overridden by features.

Show EOperations overriding Features

If an *EOperation* effectively overrides the generated code of an inherited feature, we link from the *EOperation* to the overridden feature with symbol □ .

In this case, we omit the inherited and overridden feature.

We also link to all *EOperation*s of all sub-types overridden a feature with symbol □ .

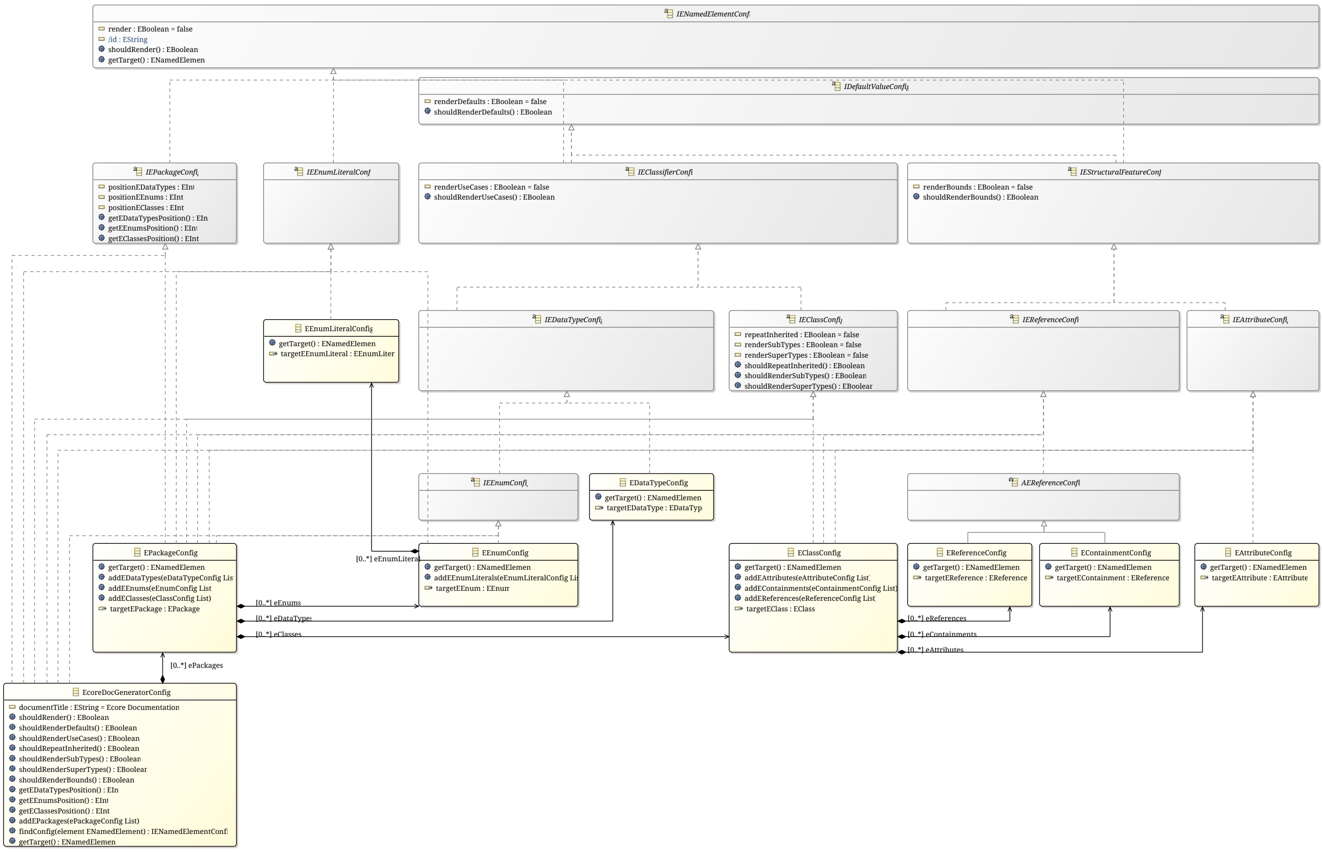
In the example, we mark both `Cl4b.getIface()` and `Cl4b.setIface()` as overriding `Cl4.iface`.
Accordingly, we mark `Cl4.iface` as being overridden by both `Cl4b.getIface()` and `Cl4b.setIface()`.
We also omit `iface` from `Cl4b` as it is effectively overridden by *EOperation*s.

Chapter 9. Generator Configuration

The generator configuration is an Ecore metamodel, so we obviously use EcoreDoc to create the documentation listed below.

The most important parts are:

- Class EcoreDocGeneratorConfig as model root, also describing the customization hierarchy
- Interface IENamedElementConfig, implemented by all elements
- Interface IDefaultValueConfig, implemented by all elements except Class EEnumLiteralConfig
- Interface IEClassifierConfig
- Interface IEClassConfig
- Interface IEPackageConfig
- Interface IEStructuralFeatureConfig


IENamedElementConf
render : EBoolean = false
/id : EString
shouldRender() : EBoolean
getTarget() : ENamedElemen
IDefaultValueConfig
renderDefaults : EBoolean = false
shouldRenderDefaults() : EBoolean
IEPackageConfi
positionEDataTypes : EInt
positionEEnums : EInt
positionEClasses : EInt
getEDataTypesPosition() : EIn
getEEnumsPosition() : EInt
getEClassesPosition() : EInt
IEEnumLiteralConf
IEClassifierConfi
renderUseCases : EBoolean = false
shouldRenderUseCases() : EBoolean
IEStructuralFeatureConf
renderBounds : EBoolean = false
shouldRenderBounds() : EBoolean
EEnumLiteralConfig
getTarget() : ENamedElemen
targetEEnumLiteral : EEnumLiter
IEDataTypeConfi
IEClassConfi
repeatInherited : EBoolean = false
renderSubTypes : EBoolean = false
renderSuperTypes : EBoolean = false
shouldRepeatInherited() : EBoolean
shouldRenderSubTypes() : EBoolean
shouldRenderSuperTypes() : EBoolear
IEReferenceConfi
IEAttributeConfi
IEEnumConfi
EDataTypeConfig
getTarget() : ENamedElemen
targetEDataType : EDataTyp
AEReferenceConfi
EPackageConfig
getTarget() : ENamedElemen
addEDataTypes(eDataTypeConfig List
addEEnums(eEnumConfig List
addEClasses(eClassConfig List)
targetEPackage : EPackage
[0..*] eEnumLiteral
EEnumConfig
getTarget() : ENamedElemen
addEEnumLiterals(eEnumLiteralConfig Li
targetEEnum : EEnum
[0..*] eEnums
[0..*] eDataTypes
[0..*] eClasses
EClassConfig
getTarget() : ENamedElemen
addEAttributes(eAttributeConfig List
addEContainments(eContainmentConfig List)
addEReferences(eReferenceConfig List
targetEClass : EClass
EReferenceConfig
getTarget() : ENamedElemen
targetEReference : EReference
EContainmentConfig
getTarget() : ENamedElemen
targetEContainment : EReference
EAttributeConfig
getTarget() : ENamedElemen
targetEAttribute : EAttribute
[0..*] eReferences
[0..*] eContainments
[0..*] eAttributes
[0..*] ePackages
EcoreDocGeneratorConfig
documentTitle : EString = Ecore Documentation
shouldRender() : EBoolean
shouldRenderDefaults() : EBoolean
shouldRenderUseCases() : EBoolean
shouldRepeatInherited() : EBoolean
shouldRenderSubTypes() : EBoolean
shouldRenderSuperTypes() : EBoolear
shouldRenderBounds() : EBoolean
getEDataTypesPosition() : EIn
getEEnumsPosition() : EInt
getEClassesPosition() : EInt
addEPackages(ePackageConfig List)
findConfig(element ENamedElement) : IENamedElementConfi
getTarget() : ENamedElemen

Generator Configuration Class Diagram

9.1. Abstract Class AEReferenceConfig

Super-types

- `config.IDefaultValueConfig`
- `config.IENamedElementConfig`
- `config.IEReferenceConfig`

- config.IEStructuralFeatureConfig
- config.IETypedElementConfig

9.2. Class EAttributeConfig

Super-types

- config.IDefaultValueConfig
- config.IEAttributeConfig
- config.IENamedElementConfig
- config.IEStructuralFeatureConfig
- config.IETypedElementConfig

References

Name	Type	Properties	Description
targetEAttribute	ecore.EAttribute	[0..1]	

Operations

Name	Aspect and Type	Properties	Description
getTarget() ▯ config.IENamedElementConfig.getTarget()	*returns* ecore.ENamedElement	[0..1]	
	`targetEAttribute`		

Used at

- config.EClassConfig.eAttributes

9.3. Class EClassConfig

Super-types

- config.IDefaultValueConfig
- config.IDiagramConfig
- config.IEAttributeConfig
- config.IEClassConfig
- config.IEClassifierConfig
- config.IENamedElementConfig
- config.IEOperationConfig
- config.IEParameterConfig
- config.IEReferenceConfig

- config.IEStructuralFeatureConfig
- config.IETypedElementConfig

Containments

Name	Type	Properties	Description
eAttributes	config.EAttributeConfig	[0..*]	
eContainments	config.EContainmentConfig	[0..*]	
eOperations	config.EOperationConfig	[0..*]	
eReferences	config.EReferenceConfig	[0..*]	

References

Name	Type	Properties	Description
targetEClass	ecore.EClass	[0..1]	

Operations

Name	Aspect and Type	Properties	Description
addEAttributes(eAttributeConfig)	*returns* void	[0..1]	Helper method for {@linkplain org.eclipse.sisu.plexus.CompositeBeanHelper#setProperty()} to handle ELists correctly.
	eAttributeConfig config.List	[0..1]	
	`EAttributes += eAttributeConfig`		
addEContainments(eContainmentConfig)	*returns* void	[0..1]	Helper method for {@linkplain org.eclipse.sisu.plexus.CompositeBeanHelper#setProperty()} to handle ELists correctly.
	eContainmentConfig config.List	[0..1]	
	`EContainments += eContainmentConfig`		

Name	Aspect and Type	Properties	Description
`addEOperations( eOperationConfig)`	*returns* `void`	`[0..1]`	Helper method for {@linkplain org.eclipse.sisu.plexus.CompositeBeanHelper#setProperty()} to handle ELists correctly.
	`eOperationConfig` `config.List`	`[0..1]`	
	`EOperations += eOperationConfig`		
`addEReferences( eReferenceConfig)`	*returns* `void`	`[0..1]`	Helper method for {@linkplain org.eclipse.sisu.plexus.CompositeBeanHelper#setProperty()} to handle ELists correctly.
	`eReferenceConfig` `config.List`	`[0..1]`	
	`EReferences += eReferenceConfig`		

`getTarget()`
`config.IENamedElementConfig.getTarget()`
	returns `ecore.ENamedElement`	`[0..1]`	

	`targetEClass`		

Used at

- `config.EPackageConfig.eClasses`

9.4. Class EContainmentConfig

Super-types

- `config.AEReferenceConfig`
- `config.IDefaultValueConfig`
- `config.IENamedElementConfig`
- `config.IEReferenceConfig`
- `config.IEStructuralFeatureConfig`
- `config.IETypedElementConfig`

References

Name	Type	Properties	Description
`targetEContainment`	`ecore.EReference`	`[0..1]`	

Operations

Name	Aspect and Type	Properties	Description
getTarget() ▯ config.IENamedElementConfig.getTarget()	*returns* ecore.ENamedElement	[0..1]	
	targetEContainment		

Used at

- config.EClassConfig.eContainments

9.5. Class EDataTypeConfig

Super-types

- config.IDefaultValueConfig
- config.IDiagramConfig
- config.IEClassifierConfig
- config.IEDataTypeConfig
- config.IENamedElementConfig

References

Name	Type	Properties	Description
targetEDataType	ecore.EDataType	[0..1]	

Operations

Name	Aspect and Type	Properties	Description
getTarget() ▯ config.IENamedElementConfig.getTarget()	*returns* ecore.ENamedElement	[0..1]	
	targetEDataType		

Used at

- config.EPackageConfig.eDataTypes

9.6. Class EEnumConfig

Super-types

- config.IDefaultValueConfig

- config.IDiagramConfig
- config.IEClassifierConfig
- config.IEDataTypeConfig
- config.IEEnumConfig
- config.IEEnumLiteralConfig
- config.IENamedElementConfig

Containments

Name	Type	Properties	Description
eEnumLiterals	config.EEnumLiteralConfig	[0..*]	

References

Name	Type	Properties	Description
targetEEnum	ecore.EEnum	[0..1]	

Operations

Name	Aspect and Type	Properties	Description
addEEnumLiterals(eEnumLiteralConfig)	*returns* void	[0..1]	Helper method for {@linkplain org.eclipse.sisu.plexus.CompositeBeanHelper#setProperty()} to handle ELists correctly.
	eEnumLiteralConfig config.List	[0..1]	
	`EEnumLiterals += eEnumLiteralConfig`		
getTarget() ▯ config.IENamedElementConfig.getTarget()	*returns* ecore.ENamedElement	[0..1]	
	`targetEEnum`		

Used at

- config.EPackageConfig.eEnums

9.7. Class EEnumLiteralConfig

Super-types

- config.IEEnumLiteralConfig

- config.IENamedElementConfig

References

Name	Type	Properties	Description
targetEEnumLiteral	ecore.EEnumLiteral	[0..1]	

Operations

Name	Aspect and Type	Properties	Description
getTarget() ▯ config.IENamedElementConfig.getTarget()	*returns* ecore.ENamedElement	[0..1]	
	`targetEEnumLiteral`		

Used at

- config.EEnumConfig.eEnumLiterals

9.8. Class EOperationConfig

Super-types

- config.AEReferenceConfig
- config.IDefaultValueConfig
- config.IENamedElementConfig
- config.IEReferenceConfig
- config.IEStructuralFeatureConfig
- config.IETypedElementConfig

Containments

Name	Type	Properties	Description
eParameters	config.EParameterConfig	[0..*]	

References

Name	Type	Properties	Description
targetEOperation	ecore.EOperation	[0..1]	

Operations

Name	Aspect and Type	Properties	Description
`addEParameters(eParameterConfig)`	*returns* `void`	`[0..1]`	Helper method for {@linkplain org.eclipse.sisu.plexus.CompositeBeanHelper#setProperty()} to handle ELists correctly.
	`eParameterConfig` `config.List`	`[0..1]`	
	`EParameters += eParameterConfig`		
`getId()` ▯ `config.IENamedElementConfig.id`	*returns* `EString`	`[0..1]`	
	`targetEOperation?.joinId`		
`getTarget()` ▯ `config.IENamedElementConfig.getTarget()`	*returns* `ecore.ENamedElement`	`[0..1]`	
	`targetEOperation`		
`joinId(eOperation)`	*returns* `EString`	`[0..1]`	
	`eOperation` `ecore.EOperation`	`[0..1]`	

	`eOperation.name +`
	`eOperation.EParameters.map[(EType?.eContainer as`
	`ENamedElement)?.name + "_" + EType?.name].join(".")`		

Used at

- `config.EClassConfig.eOperations`

9.9. Class EPackageConfig

Super-types

- `config.IDefaultValueConfig`
- `config.IDiagramConfig`
- `config.IEAttributeConfig`
- `config.IEClassConfig`
- `config.IEClassifierConfig`
- `config.IEDataTypeConfig`

- config.IEEnumConfig
- config.IEEnumLiteralConfig
- config.IENamedElementConfig
- config.IEOperationConfig
- config.IEPackageConfig
- config.IEParameterConfig
- config.IEReferenceConfig
- config.IEStructuralFeatureConfig
- config.IETypedElementConfig

Containments

Name	Type	Properties	Description
eClasses	config.EClassConfig	[0..*]	
eDataTypes	config.EDataTypeConfig	[0..*]	
eEnums	config.EEnumConfig	[0..*]	

References

Name	Type	Properties	Description
targetEPackage	ecore.EPackage	[0..1]	

Operations

Name	Aspect and Type	Properties	Description
addEClasses(eClassConfig)	*returns* void	[0..1]	Helper method for {@linkplain org.eclipse.sisu.plexus.CompositeBeanHelper#setProperty()} to handle ELists correctly.
	eClassConfig config.List	[0..1]	
	`EClasses += eClassConfig`		

Name	Aspect and Type	Properties	Description
addEDataTypes(eDataTypeConfig)	*returns* void	[0..1]	Helper method for {@linkplain org.eclipse.sisu.plexus.CompositeBeanHelper#setProperty()} to handle ELists correctly.
	eDataTypeConfig config.List	[0..1]	
	`EDataTypes += eDataTypeConfig`		
addEEnums(eEnumConfig)	*returns* void	[0..1]	Helper method for {@linkplain org.eclipse.sisu.plexus.CompositeBeanHelper#setProperty()} to handle ELists correctly.
	eEnumConfig config.List	[0..1]	
	`EEnums += eEnumConfig`		
getTarget() ▯ config. IENamedElementConfig. getTarget()	*returns* ecore. ENamedElement	[0..1]	
	`targetEPackage`		

Used at

- config.EcoreDocGeneratorConfig.ePackages

9.10. Class EParameterConfig

Super-types

- config.AEReferenceConfig
- config.IDefaultValueConfig
- config.IENamedElementConfig
- config.IEReferenceConfig
- config.IEStructuralFeatureConfig
- config.IETypedElementConfig

References

Name	Type	Properties	Description
targetEParameter	ecore.EParameter	[0..1]	

Operations

Name	Aspect and Type	Properties	Description
getTarget() □ config.IENamedElementConfig.getTarget()	*returns* ecore.ENamedElement	[0..1]	
	targetEParameter		

Used at

- config.EOperationConfig.eParameters

9.11. Class EReferenceConfig

Super-types

- config.AEReferenceConfig
- config.IDefaultValueConfig
- config.IENamedElementConfig
- config.IEReferenceConfig
- config.IEStructuralFeatureConfig
- config.IETypedElementConfig

References

Name	Type	Properties	Description
targetEReference	ecore.EReference	[0..1]	

Operations

Name	Aspect and Type	Properties	Description
getTarget() □ config.IENamedElementConfig.getTarget()	*returns* ecore.ENamedElement	[0..1]	
	targetEReference		

Used at

- config.EClassConfig.eReferences

9.12. Class EcoreDocGeneratorConfig

Root for the detailed EcoreDocGenerator configuration.

The configuration allows to specify configuration options for each element and all its contained elements. It always chooses the most specific configuration setting.

Example:

EcoreDocGeneratorConfig * renderDefaults: {unset, defaults to true} * repeatInherited: false + EPackage1 * renderDefaults: false + EClass1 + EAttribute1 * renderDefaults: true + EAttribute2 {no custom config} + EClass2 extends EClass1 + EPackage2 * repeatInherited: true + EClass3 extends EClass1 + EClass4 + EAttribute3 * renderDefaults: true * repeatInherited: false

Result:

```
EPackage1
```

```
renderDefaults false
repeatInherited false
```

```
EClass1
```

```
renderDefaults false
repeatInherited false
```

```
EAttribute1
```

```
renderDefaults true
repeatInherited false
```

```
EAttribute2
```

```
renderDefaults false
repeatInherited false
```

```
EClass2
```

```
renderDefaults false
repeatInherited false
```

```
EPackage2
```

```
renderDefaults true
repeatInherited true
```

```
EClass3
```

```
renderDefaults true
repeatInherited true
```

```
EClass4
```

```
renderDefaults true
repeatInherited true
```

```
EAttribute3
```

```
renderDefaults true
repeatInherited false
```

Super-types

- config.IDefaultValueConfig
- config.IDiagramConfig
- config.IEAttributeConfig
- config.IEClassConfig
- config.IEClassifierConfig
- config.IEDataTypeConfig
- config.IEEnumConfig
- config.IEEnumLiteralConfig
- config.IENamedElementConfig
- config.IEOperationConfig
- config.IEPackageConfig
- config.IEParameterConfig
- config.IEReferenceConfig

- config.IEStructuralFeatureConfig
- config.IETypedElementConfig

Attributes

Name	Type	Properties	Description
diagramsOutputFormat	EString	[0..1] *Default:* svg	The output-format of diagrams in the generated document. For supported outputs, see Asciidoctor Documentation. defaults to svg.
diagramsOutputPath	EString	[0..1] *Default:* .	The output path of diagrams in the generated document. defaults to the current directory.
documentTitle	EString	[0..1] *Default:* Ecore Documentation	Title of the generated document. defaults to Ecore Documentation.

Containments

Name	Type	Properties	Description
ePackages	config.EPackageConfig	[0..*]	

Operations

Name	Aspect and Type	Properties	Description
addEPackages(ePackageConfig)	*returns* void	[0..1]	Helper method for {@linkplain org.eclipse.sisu.plexus.CompositeBeanHelper#setProperty()} to handle ELists correctly.
	ePackageConfig config.List	[0..1]	
	`EPackages += ePackageConfig`		

<table>
<tr><th>Name</th><th>Aspect and Type</th><th>Properties</th><th>Description</th></tr>
<tr><td rowspan="3">findConfig(element)</td><td><em>returns</em>
config.
IENamedElementConfig</td><td>[0..1]</td><td>Convenience method to map ENamedElement → config.</td></tr>
<tr><td>element
ecore.
ENamedElement</td><td>[0..1]</td><td></td></tr>
<tr><td colspan="3"><pre><code>switch (element) {
  EPackage:
    EPackages
  EEnum:
    EPackages.flatMap[EEnums]
  EDataType:
    EPackages.flatMap[EDataTypes]
  EEnumLiteral:
    EPackages.flatMap[EEnums].flatMap[EEnumLiterals]
  EClass:
    EPackages.flatMap[EClasses]
  EAttribute:
    EPackages.flatMap[EClasses].flatMap[EAttributes]
  EReference case (element.isContainment):
    EPackages.flatMap[EClasses].flatMap[EContainments]
  EReference case (!element.isContainment):
    EPackages.flatMap[EClasses].flatMap[EReferences]
  EOperation:
    EPackages.flatMap[EClasses].flatMap[EOperations]
  EParameter:
    EPackages.flatMap[EClasses].flatMap[EOperations]
      .flatMap[EParameters]
}.findFirst[target == element]</code></pre></td></tr>
<tr><td rowspan="2">getEClassesPosition()
 config.IEPackageConfig.getEClassesPosition()</td><td><em>returns</em>
EInt</td><td>[0..1]</td><td>Sets default for positionEClasses = 3.</td></tr>
<tr><td colspan="3"><pre><code>if (isSetPositionEClasses) {
  positionEClasses
} else {
  3
}</code></pre></td></tr>
</table>

Name	Aspect and Type	Properties	Description

getEDataTypesPosition()
 	
config.IEPackageConfig.getEDataTypesPosition()
	returns EInt	[0..1]	Sets default for positionEDataTypes = 1.

	`if (isSetPositionEDataTypes) {`
	`  positionEDataTypes`
	`} else {`
	`  1`
	`}`		

getEEnumsPosition()
 	
config.IEPackageConfig.getEEnumsPosition()
	returns EInt	[0..1]	Sets default for positionEEnums = 2.

	`if (isSetPositionEEnums) {`
	`  positionEEnums`
	`} else {`
	`  2`
	`}`		

getTarget()
 	
config.IENamedElementConfig.getTarget()
	returns ecore.ENamedElement	[0..1]	

	`null`		

shouldRender()
 	
config.IENamedElementConfig.shouldRender()
	returns EBoolean	[0..1]	Sets default for render = true.

	`if (isSetRender) {`
	`  render`
	`} else {`
	`  true`
	`}`		

shouldRenderBounds()
 	
config.IETypedElementConfig.shouldRenderBounds()
	returns EBoolean	[0..1]	Sets default for renderBounds = shouldRenderDefaults().

	`if (isSetRenderBounds) {`
	`  renderBounds`
	`} else {`
	`  shouldRenderDefaults()`
	`}`		

Name	Aspect and Type	Properties	Description
`shouldRenderDefaults()` □ `config.IDefaultValueConfig.shouldRenderDefaults()`	*returns* `EBoolean`	`[0..1]`	Sets default for renderDefaults = true.

	`if (isSetRenderDefaults) {`
	`  renderDefaults`
	`} else {`
	`  true`
	`}`		

`shouldRenderDiagrams()` □ `config.IDiagramConfig.shouldRenderDiagrams()`	*returns* `EBoolean`	`[0..1]`	

	`if (isSetRenderDiagrams) {`
	`  renderDiagrams`
	`} else {`
	`  false`
	`}`		

`shouldRenderSubTypes()` □ `config.IEClassConfig.shouldRenderSubTypes()`	*returns* `EBoolean`	`[0..1]`	Sets default for renderSubTypes = true.

	`if (isSetRenderSubTypes) {`
	`  renderSubTypes`
	`} else {`
	`  true`
	`}`		

`shouldRenderSuperTypes()` □ `config.IEClassConfig.shouldRenderSuperTypes()`	*returns* `EBoolean`	`[0..1]`	Sets default for RenderSuperTypes = true.

	`if (isSetRenderSuperTypes) {`
	`  renderSuperTypes`
	`} else {`
	`  true`
	`}`		

`shouldRenderUseCases()` □ `config.IEClassifierConfig.shouldRenderUseCases()`	*returns* `EBoolean`	`[0..1]`	Sets default for renderUseCases = true.

	`if (isSetRenderUseCases) {`
	`  renderUseCases`
	`} else {`
	`  true`
	`}`		

<table>
<tr><th>Name</th><th>Aspect and Type</th><th>Properties</th><th>Description</th></tr>
<tr><td rowspan="2">shouldRepeatInherited()

 config.IEClassConfig.
shouldRepeatInherited()</td><td><em>returns</em>
EBoolean</td><td>[0..1]</td><td>Sets default for repeatInherited = true.</td></tr>
<tr><td colspan="3"><pre>if (isSetRepeatInherited) {
  repeatInherited
} else {
  true
}</pre></td></tr>
</table>

9.13. Interface IDefaultValueConfig

Attributes

Name	Type	Properties	Description
renderDefaults	EBoolean	[0..1] unsettable	Whether properties should be rendered at their default values. Example: If EReference.ordered = true (the default value), the ordered property of that EReference will not be rendered if renderDefaults = false.

Operations

<table>
<tr><th>Name</th><th>Aspect and Type</th><th>Properties</th><th>Description</th></tr>
<tr><td rowspan="2">shouldRenderDefaults()

 config.
EcoreDocGeneratorConfig.
shouldRenderDefaults()</td><td><em>returns</em>
EBoolean</td><td>[0..1]</td><td>Traverses the tree to find the most specific renderDefaults setting.</td></tr>
<tr><td colspan="3"><pre>if (isSetRenderDefaults) {
  renderDefaults
} else {
  (eContainer as IDefaultValueConfig).shouldRenderDefaults()
}</pre></td></tr>
</table>

9.14. Interface IDiagramConfig

Attributes

Name	Type	Properties	Description
renderDiagrams	EBoolean	[0..1] unsettable	Whether diagrams should be rendered.

Operations

<table>
<tr><th>Name</th><th>Aspect and Type</th><th>Properties</th><th>Description</th></tr>
<tr><td rowspan="2">shouldRenderDiagrams()
□ config.EcoreDocGeneratorConfig.shouldRenderDiagrams()</td><td><i>returns</i> EBoolean</td><td>[0..1]</td><td>Traverses the tree to find the most specific renderDiagrams setting.</td></tr>
<tr><td colspan="3"><pre>if (isSetRenderDiagrams) {
  renderDiagrams
} else {
  (eContainer as IDiagramConfig).shouldRenderDiagrams()
}</pre></td></tr>
</table>

9.15. Interface IEAttributeConfig

Super-types

- config.IDefaultValueConfig
- config.IENamedElementConfig
- config.IEStructuralFeatureConfig
- config.IETypedElementConfig

9.16. Interface IEClassConfig

Super-types

- config.IDefaultValueConfig
- config.IDiagramConfig
- config.IEClassifierConfig
- config.IENamedElementConfig

Attributes

Name	Type	Properties	Description
renderSubTypes	EBoolean	[0..1] unsettable	Whether the list of sub-types should be rendered.
renderSuperTypes	EBoolean	[0..1] unsettable	Whether the list of super-types should be rendered.

Name	Type	Properties	Description
repeatInherited	EBoolean	[0..1] unsettable	Whether inherited features should be repeated. Example: EClass1 has an EAttribute name=attr1. EClass2 extends EClass1. If repeatInherited = true for EClass2, attr1 will be listed in the section of EClass1 and EClass2. Otherwise, attr1 will only be listed in the section of EClass1.

Operations

Name	Aspect and Type	Properties	Description

shouldRenderSubTypes()
config.EcoreDocGeneratorConfig.shouldRenderSubTypes()
	returns EBoolean	[0..1]	Traverses the tree to find the most specific renderSubTypes setting.

	`if (isSetRenderSubTypes) {`
	`  renderSubTypes`
	`} else {`
	`  (eContainer as IEClassConfig).shouldRenderSubTypes()`
	`}`		

shouldRenderSuperTypes()
config.EcoreDocGeneratorConfig.shouldRenderSuperTypes()
	returns EBoolean	[0..1]	Traverses the tree to find the most specific renderSuperTypes setting.

	`if (isSetRenderSuperTypes) {`
	`  renderSuperTypes`
	`} else {`
	`  (eContainer as IEClassConfig).shouldRenderSuperTypes()`
	`}`		

shouldRepeatInherited()
config.EcoreDocGeneratorConfig.shouldRepeatInherited()
	returns EBoolean	[0..1]	Traverses the tree to find the most specific repeatInherited setting.

	`if (isSetRepeatInherited) {`
	`  repeatInherited`
	`} else {`
	`  (eContainer as IEClassConfig).shouldRepeatInherited()`
	`}`		

9.17. Interface IEClassifierConfig

Super-types

- config.IDefaultValueConfig

- config.IDiagramConfig
- config.IENamedElementConfig

Attributes

Name	Type	Properties	Description
renderUseCases	EBoolean	[0..1] unsettable	Whether use cases (references to other usages of this element) should be rendered.

Operations

Name	Aspect and Type	Properties	Description
shouldRenderUseCases() □ config.EcoreDocGeneratorConfig.shouldRenderUseCases()	*returns* EBoolean	[0..1]	Traverses the tree to find the most specific renderUseCases setting.

```
if (isSetRenderUseCases) {
  renderUseCases
} else {
  (eContainer as IEClassifierConfig).shouldRenderUseCases()
}
```

9.18. Interface IEDataTypeConfig

Super-types

- config.IDefaultValueConfig
- config.IDiagramConfig
- config.IEClassifierConfig
- config.IENamedElementConfig

9.19. Interface IEEnumConfig

Super-types

- config.IDefaultValueConfig
- config.IDiagramConfig
- config.IEClassifierConfig
- config.IEDataTypeConfig
- config.IENamedElementConfig

9.20. Interface IEEnumLiteralConfig

Super-types

- `config.IENamedElementConfig`

9.21. Interface IENamedElementConfig

Attributes

Name	Type	Properties	Description
`render`	`EBoolean`	`[0..1]` unsettable	Whether this element should be rendered at all.

Operations

Name	Aspect and Type	Properties	Description
abstract getTarget() ▯ config.EAttributeConfig.getTarget() ▯ config.EClassConfig.getTarget() ▯ config.EContainmentConfig.getTarget() ▯ config.EDataTypeConfig.getTarget() ▯ config.EEnumConfig.getTarget() ▯ config.EEnumLiteralConfig.getTarget() ▯ config.EOperationConfig.getTarget() ▯ config.EPackageConfig.getTarget() ▯ config.EParameterConfig.getTarget() ▯ config.EReferenceConfig.getTarget() ▯ config.EcoreDocGeneratorConfig.getTarget()	*returns* ecore.ENamedElement	[0..1]	Link to the configured element.

<table>
<tr><th>Name</th><th>Aspect and Type</th><th>Properties</th><th>Description</th></tr>
<tr><td rowspan="2">shouldRender()
▯ config.EcoreDocGeneratorConfig.shouldRender()</td><td><em>returns</em> EBoolean</td><td>[0..1]</td><td>Traverses the tree to find the most specific render setting.</td></tr>
<tr><td colspan="3">if (isSetRender) {
  render
} else {
  (eContainer as IENamedElementConfig).shouldRender()
}</td></tr>
</table>

Used at

- config.EcoreDocGeneratorConfig.findConfig(element)

9.22. Interface IEOperationConfig

Super-types

- config.IDefaultValueConfig
- config.IENamedElementConfig
- config.IETypedElementConfig

9.23. Interface IEPackageConfig

Super-types

- config.IDiagramConfig
- config.IENamedElementConfig

Attributes

Name	Type	Properties	Description
positionEClasses	EInt	[0..1] unsettable	Rendering position of all EClasses within an EPackage.
positionEDataTypes	EInt	[0..1] unsettable	Rendering position of all EDataTypes within an EPackage.
positionEEnums	EInt	[0..1] unsettable	Rendering position of all EEnums within an EPackage.

Operations

<table>
<tr><th>Name</th><th>Aspect and Type</th><th>Properties</th><th>Description</th></tr>
<tr><td rowspan="2">getEClassesPosition()
□ config.
EcoreDocGeneratorConfig.
getEClassesPosition()</td><td><em>returns</em>
EInt</td><td>[0..1]</td><td>Traverses the tree to find the most specific positionEClasses setting.</td></tr>
<tr><td colspan="3">if (isSetPositionEClasses) {
  positionEClasses
} else {
  (eContainer as IEPackageConfig).getEClassesPosition()
}</td></tr>
<tr><td rowspan="2">getEDataTypesPosition()
□ config.
EcoreDocGeneratorConfig.
getEDataTypesPosition()</td><td><em>returns</em>
EInt</td><td>[0..1]</td><td>Traverses the tree to find the most specific positionEDataTypes setting.</td></tr>
<tr><td colspan="3">if (isSetPositionEDataTypes) {
  positionEDataTypes
} else {
  (eContainer as IEPackageConfig).getEDataTypesPosition()
}</td></tr>
<tr><td rowspan="2">getEEnumsPosition()
□ config.
EcoreDocGeneratorConfig.
getEEnumsPosition()</td><td><em>returns</em>
EInt</td><td>[0..1]</td><td>Traverses the tree to find the most specific positionEEnums setting.</td></tr>
<tr><td colspan="3">if (isSetPositionEEnums) {
  positionEEnums
} else {
  (eContainer as IEPackageConfig).getEEnumsPosition()
}</td></tr>
</table>

9.24. Interface IEParameterConfig

Super-types

- config.IDefaultValueConfig
- config.IENamedElementConfig
- config.IETypedElementConfig

9.25. Interface IEReferenceConfig

Super-types

- config.IDefaultValueConfig
- config.IENamedElementConfig
- config.IEStructuralFeatureConfig
- config.IETypedElementConfig

9.26. Interface IEStructuralFeatureConfig

Super-types

- config.IDefaultValueConfig
- config.IENamedElementConfig
- config.IETypedElementConfig

9.27. Interface IETypedElementConfig

Super-types

- config.IDefaultValueConfig
- config.IENamedElementConfig

Attributes

Name	Type	Properties	Description
renderBounds	EBoolean	[0..1] unsettable	Whether multiplicity bounds should be rendered, even if they are at their default values and renderDefaults = false.

Operations

Name	Aspect and Type	Properties	Description
shouldRenderBounds() ▯ config.EcoreDocGeneratorConfig.shouldRenderBounds()	*returns* EBoolean	[0..1]	Traverses the tree to find the most specific renderBounds setting.

```
if (isSetRenderBounds) {
  renderBounds
} else if(isSetRenderDefaults) {
  renderDefaults
} else {
  (eContainer as IETypedElementConfig).shouldRenderBounds()
}
```

Chapter 10. Versions

This asset in version 2024-11-26 15:12 UTC was developed using the following components and versions.

Eclipse	4.16 (2020-06)
Google Guava	27.1
Apache Commons Lang3	3.10
Apache Commons IO	2.2
Apache Maven	3.3.9
Eclipse Ecore	2.22.0
Eclipse Xcore	1.14.0
Eclipse Tycho	1.7.0

Chapter 11. Known Issues

- If HTML is used in Ecore documentation, the PDF rendering can be faulty (Issue #2)
- `EAnnotation`s are missing from the documentation (Issue #3)