yWorks UML Doclet 3.1 User's Guide

Contents

1. Introduction

Welcome to the *yWorks UML Doclet User's Guide*.
This guide explains how to use *yWorks UML Doclet* or ***yDoc*** for short, a javadoc extension (more specifically a doclet/taglet bundle) that provides

- functionality to auto-generate, customize, and include UML diagrams in the API documentation of your Java products
- a filter interface which allows for custom suppression of class, field, or method documentation
- an easy to use mechanism for defining simple custom tags via XML

Although, yDoc is designed in such a way that it allows user's to continue using all the features they know from standard javadoc, some basic knowledge about javadoc usage in general and doclet usage specifically is required to successfully use yDoc.

Detailed information on javadoc is available at
https://docs.oracle.com/javase/8/docs/technotes/guides/javadoc/index.html.
Detailed information on doclets is available at
https://docs.oracle.com/javase/8/docs/technotes/guides/javadoc/doclet/overview.html.

2. System Requirements

yDoc 3.1 requires JDK 1.8 installed on your system.
To view UML class diagrams in SVG or SVGZ format, you need a browser with native SVG support. To view UML class diagrams in SWF format, you need a browser with a Flash Player plug-in. Flash Player plug-ins are available from Adobe.
If you want to run yDoc under Unix/Linux operating systems, you need to have an X server installed and running, since yDoc makes use of the java awt and/or swing packages (for UML generation only).

3. Installing yDoc

Unzip the yDoc archive (yworks-uml-doclet-3.1-jdk1.8.zip) into a directory of your choice. It will create a lib/, a doc/, and a resources/ subdirectory.
The lib directory contains the java classes you need to run the ydoc expansion as jar libraries.
The doc directory contains the yDoc User's Guide in HTML and PDF format, the DocFilter and PathResolver API Documentation in HTML format, and several usage samples.
The resources directory contains various configuration files which you can use to customize the behaviour of the yDoc expansion. See Configuring yDoc and Using the XML driven taglet factory for more details.

4. Running yDoc

Basically you run javadoc. The only difference is, that you tell javadoc to use the facilities provided by yDoc as a plug-in.
Read on for detailed information on how to do that. You can skip this part, if you are already familiar with using custom doclets for javadoc.

Using the yDoc doclet

We recommend running javadoc either using a build tool such as ANT (version 1.5.2 or better) or directly from commandline. Before running javadoc from commandline, put your commandline options into a file called "options" and run javadoc by invoking
  javadoc @options @packages
where "packages" is the filename of a file containing the java packages you want to be documented.

See yDoc Quick Start for simple examples on how to use yDoc.

For detailed documentation on the javadoc options, see the javadoc tool homepage at https://docs.oracle.com/javase/8/docs/technotes/guides/javadoc/index.html.

To use the yDoc expansion the following options are especially important:

- **-docletpath** *docletpathlist*
  This option tells javadoc where to look for the yDoc expansion.
  The *docletpathlist* must contain the path to the library ydoc.jar
  `<ydoc_install_dir>/lib/ydoc.jar`
  and the resources directory
  `<ydoc_install_dir>/resources`
  Important:
  If you want to use the yDoc UML generation, *docletpathlist* must also contain the path to your *compiled*, *unobfuscated* Java class files (*.class), for which you want to generate the API documentation, and to all libraries needed to compile your Java source files.
- **-doclet ydoc.doclets.YStandard**
  The -doclet ydoc.doclets.YStandard option finally tells javadoc to actually use the YStandard doclet, which is the core class of the ydoc expansion.

See yDoc Features for information on (custom) commandline options and on how to use the specific capabilites of yDoc.

A sample options file on a Win32 operating system could look like this:

```
-d <destination directory>
-sourcepath <source directory>
-breakiterator
-generic
-umlautogen
-author
-docletpath <YID>/lib/ydoc.jar;<YID>/resources;<some path>/myapp.jar
-doclet ydoc.doclets.YStandard
-filterpath <YID>/lib/ydoc.jar
-filter ydoc.filters.ExcludeFilter
-tagletpath <YID>/lib/ydoc.jar
-tag param
-tag return
```

```
  -tag see
  -ytag y.uml
```

where **`<YID>`** denotes the **`<ydoc_install_dir>`**.
On Unix/Linux operating systems, you will have to use " **`:`** " as a path separator instead of " **`;`** ".

Using the yDoc doclet from within a Java IDE

Running yDoc from within Eclipse 3

1. Select *Export* from the *File* menu.

2. Choose *Javadoc* from *Select an export destination*.
   Go to the next tab.

3. Select *Use Custom Doclet* then specify *Doclet name* and *Doclet class path*.

   Name has to be ydoc.doclets.YStandard and path has to be **`<yid>/lib/ydoc.jar`**.
   `<yid>` denotes the absolute path to the yDoc directory.

   Go to the next tab.

4. Add in *Extra Javadoc options*
   -docletpath **`<yid>/resources`**

   If your sources depend on additional libraries, you also need to append the path to these libraries to the above line.

   Any other options you want to use, e.g. -d **`<destination>`** or -umlautogen, need to be specified in this input area, too.

5. Optionally, add -J-Xmx1024m in *VM options*.
   You may want to play with the numerical value depending on available RAM and project size.

Running yDoc from within IntelliJ Idea 6

1. Select *Generate JavaDoc ...* from the *Tools* menu.

2. Add in *Other command line arguments*
   -docletpath **`"<yid>/lib/ydoc.jar"`** -doclet **`ydoc.doclets.YStandard`** -resourcepath **`"<yid>/resources"`**
   `<yid>` denotes the absolute path to the yDoc directory.

   If your sources depend on additional libraries, you also need to append the path to these libraries to the -docletpath option.

   Any other options you want to use, e.g. -d **`<destination>`** or -umlautogen, need to be specified in this input field, too.

3. Click *Start*.

Running yDoc from within Netbeans 5.5

1. Switch to *Projects* view.

2. Open context menu for *Source Packages* (by right clicking).
   Choose *Properties*.

3. Expand *Build*.
   Choose *Documenting*.

4. Add in *Additional Javadoc Options*
   -docletpath **`"<yid>/lib/ydoc.jar"`** -doclet **`ydoc.doclets.YStandard`** -resourcepath **`"<yid>/resources"`**
   `<yid>` denotes the absolute path to the yDoc directory.

   If your sources depend on additional libraries, you also need to append the path to these libraries to the -docletpath option.

   Any other options you want to use, e.g. -d **`<destination>`** or -umlautogen, need to be specified in this input field, too.

5. Choose *Generate Javadoc for "<project name>"* from the *Build* menu.

yDoc Quick Start

This section demonstrates how to use yDoc to generate a Javadoc page of a sample class that will automatically include an UML diagram depicting that class.

- **yDoc from commandline**

  Look in **`<YID>/doc/examples`** for sample options files and sample Java sources to test yDoc.
  All you need to do is invoking javadoc in **`<ydoc_install_dir>`** with either
  javadoc @doc/examples/options.sample.linux
  or
  javadoc @doc/examples/options.sample.win32
  depending on your operating system.

- **yDoc in ANT**

  Using ANT 1.5.2 or better, you can use ANT's javadoc task to run yDoc.

  Look in **`<YID>/doc/examples`** for a sample ANT build file and sample Java sources to test yDoc.
  All you need to do is invoking ant in **`<ydoc_install_dir>`** with
  ant -buildfile doc/examples/build-sample.xml test-ydoc

The generated API pages can now be found in

`<ydoc_install_dir>/doc/api/examples`.

Note that the generated UML diagrams are in PNG format. If you want to generate the uml diagrams in a different format (SVG, SVGZ, SWF, GIF, JPG) simply change the value of **formats.fileformat** in the yDoc configuration file **`<ydoc_install_dir>/resources/ydoc.cfg`** accordingly.

The next tutorial step would be to look in **`<ydoc_install_dir>/doc/examples`** for the sample option/build files and sample Java sources which have been used in this example. Once you understand the options and tags, you are ready to use yDoc in your own project.

5. yDoc Features

Generating UML class diagrams

yDoc will generate UML diagrams, if one or more of the following commandline options are used:

- **-umlgen**
- **-umltypegen**
- **-umlpackagegen**
- **-umloverviewgen**
- **-umlautogen**

All UML diagrams feature hyperlinks for the displayed packages, types, and type members, which allow direct access to the corresponding documentation.

See Custom command line options for additional details.

yDoc supports several output formats for UML diagrams, including SVG, SWF, and PNG and will automatically integrate the diagrams into the generated HTML API documentation.
Moreover, yDoc provides many settings which allow to customize the generated UML diagrams in great detail.

See Configuring yDoc for details.

Important:
yDoc uses the Java Reflection API to generate the class diagrams, therefore you need to specify the path to your *compiled, unobfuscated* Java class files (*.class) and to all libraries needed to compile your Java source files in the -docletpath option.
Your class files may be located in a jar file.

Alternatively, yDoc can embed predefined diagrams instead of generating them. Predefined diagrams have to be available in GraphML. GraphML is a generic graph interchange file format. Diagrams in this format can, e.g., be created using yEd, yWorks' free graph editor.

For yDoc to be able to find and read such diagrams, a diagram locator has to be specified.

The distribution comes with one predefined diagram locator. For this default locator to work successfully, predefined diagram files have to meet the following criteria:

- Overview diagrams must be in the same directory as your overview.html. The diagram files have to be named **`overview <id> .graphml`**, where **`<id>`** denotes the diagram ID (see also Configuring yDoc).
- Package diagrams must be in the same directory as your package.html. The diagram files have to be named **`package <id> .graphml`**, where **`<id>`** denotes the diagram ID (see also Configuring yDoc).
- Type diagrams must be in the same directory as the corresponding source file. The diagram files have to follow the same naming convention as the Java source file except for the **`.graphml`** file extension.

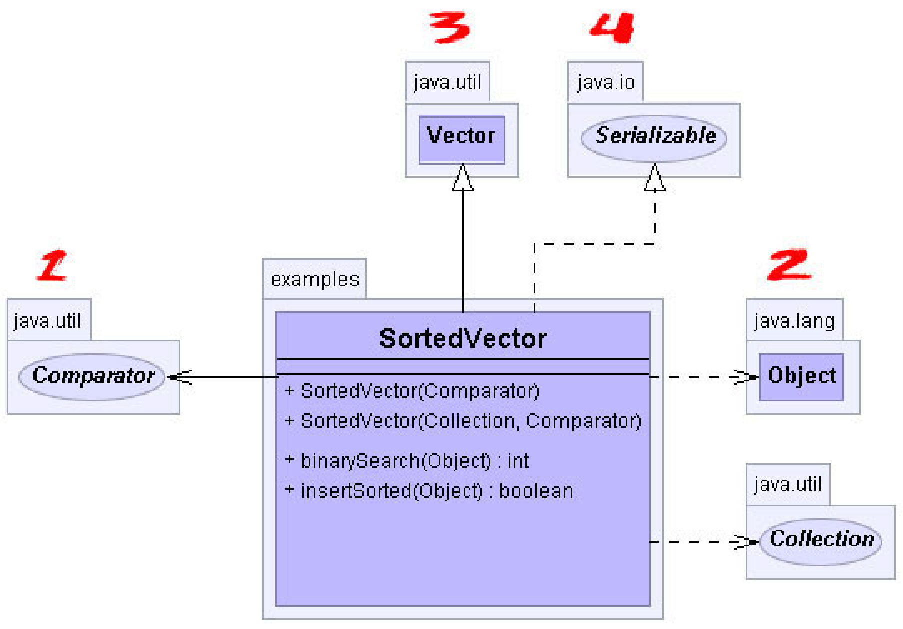
work successfully, predefined diagram files have to meet the following criteria:

To use this default diagram locator, you need to specify the following two commandline options:
  -diagramlocatorpath **`<ydoc_install_dir>/lib/ydoc.jar`**
  -diagramlocator ydoc.resolvers.DefaultDiagramLocator

To create and use your own diagram locators, all you have to do is implementing the ydoc.resolvers.PathResolver interface and register the locator similar to the above example. The mechanism to register locators works similar to the one used for doclets.

PathResolver API
Documentation for the ydoc.resolvers.PathResolver interface, which comprises the PathResolver API.

General Layout of UML class diagrams


3
4
java.util
Vector
java.io
Serializable
1
2
examples
SortedVector
+ SortedVector(Comparator)
+ SortedVector(Collection, Comparator)
+ binarySearch(Object) : int
+ insertSorted(Object) : boolean
java.util
Comparator
java.lang
Object
java.util
Collection

1 **Associations**
    structural relationships between a whole and its parts, i.e. *has a* or *instantiates*
  Every declared field constitutes an association.

2 **Dependencies**
    semantic relationships in which a change to one thing may effect the semantics of the other thing
  There are several heuristics as to what constitues a dependency.
  See Configuring yDoc.

3 **Generalizations**
specialization/generalization relationships, i.e. *is a* or *subclass/superclass*
For interfaces there may be more than one generalization relationship.

4 **Realizations**
semantic relationships between classifiers, i.e. *interface/implementing class*
For interfaces there are no realization relationships.

Using filters

yDoc provides a sophisticated filter framework, which lets you exclude parts of your API from documentation using customizable filter criteria.

The distribution comes with one predefined filter, which lets you exclude classes/interfaces, fields, and/or methods from documentation, if their documentation comment contains an **@y.exclude** tag.
To use this filter, you need to specify the following two commandline options:
-filterpath **`<ydoc_install_dir>/lib/ydoc.jar`**
-filter ydoc.filters.ExcludeFilter

To create and use your own filters, all you have to do is implement the ydoc.filters.DocFilter interface and register the filter similar to the above example.
The mechanism to register filters works similar to the one used for doclets.

DocFilter API
Documentation for the ydoc.filters.DocFilter interface, which comprises the DocFilter API.

Using the XML driven taglet factory

By specifying the -generic option, you can tell yDoc to register simple taglets, which are more powerful than the ones created by the standard -tag option and are defined in the **`resources/taglet_definitions.xml`** and **`resources/taglet_templates.xml`** files.
By adding more definitions to those files, you can use/register more simple taglets.

The basic idea is to have template definitions that define taglet behaviour and taglet definitions that define scope, name, and which template to use.

For examples on how to define taglets, see the two mentioned xml files.

Taglet Definitions
The following XML elements are used to define taglets:

- `<taglet>`

Each of these elements results in the registration of one particular taglet.
The value of the `name` attribute specifies the javadoc tag for the taglet. The value of the attribute `allowMultipleTags` specifies if more than one appearance of the javadoc tag per doc element is allowed. If not, all but the first tag will be ignored.

- `<usage>`
  Required element that specifies the taglet scope as per the taglet API.
- `<headline>`
  Required root element for `<singular>` and `<plural>`.
- `<singular>`
  Required element that specifies the headline for the tag comment if only one javadoc tag or no `<plural>` element is present.
- `<plural>`
  Optional element that specifies the headline for the tag comment if multiple javadoc tags are allowed and present.

In general, it is a good idea to use at least one '.' character in the name of custom tags to avoid potential conflicts/overrides.

Template Definitions
The following XML elements are used to define templates:

- `<template>`
  Each of these elements results in the creation of one particular template.
  The value of the `name` attribute has to be unique among all templates. It is used to reference the template in the taglet definition.
- `<headline>`
  Required element that specifies the HTML code for the headline of the tag comment.
  You may specify one parameter sign, i.e. `#0`.
  You may use a single parameter multiple times, e.g. `<headline>` `<![CDATA[bla #0 bla#0bla]]></headline>`
  The element should contain unparsed character data, i.e. `<![CDATA[....]]>`
- `<content>`
  Required element that specifies the HTML formatting for the tag comment.
  The value of the `separator` attribute specifies if and how to break down the comment into parameters.
  Possible values are:

- any single character	breaks the comment at each occurrence of the specified character
- the token "first-whitespace"	breaks the comment at the first occurrence of a whitespace
- the token "first-word-or-tag"	breaks the comment at the first occurrence of a whitespace that does not belong to an inline tag
- the token "whitespace"	breaks the comment at each occurrence of a whitespace
- the token "none" (default)	results in one token only, namely the whole comment

- `<content-item>`
  Required element that specifies the HTML code to wrap the tag comment in. If multiple javadoc tag are present for a particular doc element, then one content

item is created for each tag comment.
You may specify up to ten parameter signs, i.e. #x, where -1 < x < 10.
You may use a single parameter multiple times.
The element should contain unparsed character data.

- `<content-sep>`
  Optional element that defaults to "".
  Its value will be inserted between content items.
  The element should contain unparsed character data.
- `<content-start>`
  Optional element that defaults to "".
  Its value will be inserted directly after the headline, before the first content item.
  The element should contain unparsed character data.
- `<content-end>`
  Optional element that defaults to "".
  Its value will be inserted directly after the the last content item.
  The element should contain unparsed character data.

In general, it is a good idea to use the `<DT>` tag for headlines and the `<DD>` tag for content, since all output generated by javadoc taglets appears in definition lists.

Custom command line options

yDoc provides several custom command line options:

- **-diagramlocator** *class*
  Specifies the class file for the diagram locator to be used. Use the fully-qualified name for *class*. Use the -diagramlocatorpath option to specify the path to the diagram locator.
- **-diagramlocatorpath** *-diagramlocatorpathlist*
  Specifies the search paths for finding diagram locator class files (*.class). The *diagramlocatorpathlist* can contain multiple paths separated by the system-dependant path-separator.
- **-filter** *class*
  Specifies the class file for the filter to be applied. Use the fully-qualified name for *class*. Use the -filterpath option to specify the path to the filter.
- **-filterpath** *filterpathlist*
  Specifies the search paths for finding filter class files (*.class). The *filterpathlist* can contain multiple paths separated by the system-dependant path-separator.
- **-generic**
  The taglet definitions in **`resources/taglet_definitions.xml`** and **`resources/taglet_templates.xml`** are used to create and register simple taglets.
- **-license** *file*
  Specifies the path to the license file.
- **-resourcepath** *resourcepathlist*
  Specifies the search paths for finding resource files (i.e. taglet definition files, ydoc configuration file, ydoc license file). The *resourcepathlist* can contain multiple paths separated by the system-dependant path-separator.
- **-umlautogen**

Same as using -umltypegen, -umlpackagegen, and -umloverviewgen in combination

- **-umlfileformat** *formatname*
  Overrides the uml_file_format property in **`resources/ydoc.cfg`**
  See the section about UML file formats for a list of supported formats.
- **-umlgen**
  UML diagrams will be created and embedded for all documented files with an **@y.uml** tag.
  @y.uml may be used in type, package, and overview documentation.
- **-umloverviewgen**
  An UML overview diagram will be created and embedded, even if there is no **@y.uml** tag in overview.html.
- **-umlpackagegen**
  UML diagrams will be created and embedded for all documented packages, not only for those with an **@y.uml** tag.
- **-umltypegen**
  UML diagrams will be created and embedded for all documented classes and interfaces, not only for those with an **@y.uml** tag.
- **-ytag**
  Allows to specify the position of custom yDoc tags (i.e. @y.uml or tags defined via the taglet factory) in relation to standard tags.

6. Configuring yDoc

resources/ydoc.cfg

`resources/ydoc.cfg` is yDoc's main configuration file.
It uses a simple XML format consisting of nested group and property elements.

Following is the complete list of recognized group and property declarations.

Group **diagrams**

Encapsulates settings that determine all aspects of yDoc's UML generating mechanism. These settings are grouped into the categories *overview*, *package*, and *type* in correspondance to the available diagram types.
You may have multiple *diagram* subgroups in each of the above mentioned three categories. For each *diagram* group, one UML diagram will be generated and embedded into the corresponding HTML file.

Group **diagrams.overview.diagram**

- Property style accepts an arbitrary text value interpreted as a file name.
  This property specifies the path to a yDoc style definition file, which determines the visual properties of the generated UML diagram, such as line colors or font sizes.
  See UML Styles for more information.
- Property type accepts one of the following values:
  - dependency
    Dependency diagrams depict package-level dependencies in your project.
  - inheritance
    Inheritance diagrams depict project-wide inheritance trees.

  This property specifies the overview diagram type.
- Property id accepts an arbitrary text value.
  This property specifies a diagram ID, which is used to distinguish between multiple overview diagrams.
  The value of this property has to be unique among all diagrams.overview.diagram ID values.

Group **diagrams.overview.diagram.include**

- Property dependencies accepts one of the following values:
  - all
    All package dependencies are displayed.
  - reduced

No transitive dependencies are displayed.

This property specifies whether transitive dependencies should be displayed.
This property is only respected for overview diagrams of type dependency.

- Property groups accepts values true and false.
This property specifies whether package nodes should be grouped according to the -group options. If no -group option is used, this property is ignored.
- Property packages accepts values true and false.
This property specifies whether type nodes should be grouped according to their containing packages.
This property is only respected for overview diagrams of type inheritance.

Group **diagrams.overview.diagram.insets**

- Property group accepts non-negative integer values.
This property specifies the distance from a group node's border to the package nodes contained in the group node.
- Property package accepts non-negative integer values.
This property specifies the distance from a package node's border to the type nodes contained in the package node.

Group **diagrams.overview.diagram.layout**

- Property BUS_ROUTING accepts values true and false.
This property specifies whether multiple relations (e.g. in the case of a class having multiple subclasses) should be routed in a bus style manner.
- Property CYCLE_LAYERING_POLICY accepts one of the following values:
  - DEFAULT_POLICY
  All backwards relation edges as determined by a depth-first-search on the diagram nodes are temporarily removed for layering.
  - ASSIGN_CYCLES_TO_SAME_LAYER_POLICY
  Diagram nodes with cyclic dependencies are put into the same layer.
  - BREAK_CYCLES_BY_WEIGHT_POLICY
  Cyclic dependencies are resolved by temporarily removing the least significant relation edges for layering.

  This property specifies the layering policy for cyclic dependencies.
- Property GROUP_COMPACTION accepts values true and false.
This property specifies whether package and group digram nodes should be kept as small as possible.
- Property ORIENTATION accepts one of the following values:

  - TOP_TO_BOTTOM
  - LEFT_TO_RIGHT
  - RIGHT_TO_LEFT
  - BOTTOM_TO_TOP

  This property specifies the layout orientation.
- Property RECURSIVE_GROUP_LAYERING accepts values true and false.
  This property specifies whether layering should be performed locally on a per group basis or globally for the whole diagram.
  This property is ignored, if the diagram does not contain node groups.
- Property REVERSE_EDGES accepts values true and false.
  This property specifies whether the direction of relation edges should be reversed during layout calculation.
  Reversing the edge directions does e.g. affect the alignment of the diagram nodes.
- Property ROUTE_ORTHOGONAL accepts values true and false.
  This property specifies whether relation edges should be routed orthogonally or polyline-style.

Group **diagrams.package.diagram**

- Property style accepts an arbitrary text value interpreted as a file name.
  This property specifies the path to a yDoc style definition file, which determines the visual properties of the generated UML diagram, such as line colors or font sizes.
  See UML Styles for more information.

Group **diagrams.package.diagram.include**

- Property packages accepts values true and false.
  This property specifies whether type nodes should be grouped according to their containing package.

Group **diagrams.package.diagram.insets**

- Property group accepts non-negative integer values.
  This property specifies the distance from a group node's border to the package nodes contained in the group node.
- Property package accepts non-negative integer values.
  This property specifies the distance from a package node's border to the type nodes contained in the package node.

Group **diagrams.package.diagram.layout**

- Property BUS_ROUTING accepts values true and false.
  This property specifies whether multiple relations (e.g. in the

case of a class having multiple subclasses) should be routed in a bus style manner.

- Property ORIENTATION accepts one of the following values:
  - TOP_TO_BOTTOM
  - LEFT_TO_RIGHT
  - RIGHT_TO_LEFT
  - BOTTOM_TO_TOP

  This property specifies the layout orientation.
- Property REVERSE_EDGES accepts values true and false.
  This property specifies whether the direction of relation edges should be reversed during layout calculation.
  Reversing the edge directions does e.g. affect the alignment of the diagram nodes.
- Property ROUTE_ORTHOGONAL accepts values true and false.
  This property specifies whether relation edges should be routed orthogonally or polyline-style.

Group **diagrams.type.diagram**

- Property style accepts an arbitrary text value interpreted as a file name.
  This property specifies the path to a yDoc style definition file, which determines the visual properties of the generated UML diagram, such as line colors or font sizes.
  See UML Styles for more information.

Group **diagrams.type.diagram.exclude.pattern**

- Property associations accepts a pattern text value.
  Type names matching the pattern text will not be displayed among the diagram's association types.
- Property dependencies accepts a pattern text value.
  Type names matching the pattern text will not be displayed among the diagram's dependency types.
- Property generalizations accepts a pattern text value.
  Type names matching the pattern text will not be displayed among the diagram's generalization types.
- Property realizations accepts a pattern text value.
  Type names matching the pattern text will not be displayed among the diagram's realization types.

Pattern text values are a comma-separated list of full-qualified typename patterns where the '?' character denotes a wildcard of length one and the '*' character denotes a wildcard of arbitrary length.

Group **diagrams.type.diagram.include**

- Property associations accepts values true and false.
  This property specifies whether association nodes should be

displayed.

- Property dependencies accepts one of the following values:
    - all
      Any non-primitive type referenced in a type's byte code is considered a dependency.
    - none
      No dependency information is calculated.
    - parameters
      Any non-primitive parameter types of constructor and method signatures are considered dependencies.
    - parameters-returntype
      Any non-primitive parameter types of constructor and method signatures as well as all non-primitive method return types are considered dependencies.

  This property specifies the heuristic approach as to what constitutes a dependency. Note, that types which are associations will not appear as dependencies no matter which heuristic is chosen.
- Property packages accepts values true and false.
  This property specifies whether type nodes should be grouped according to their containing package.
- Property paramters accepts values true and false.
  This property specifies whether parameter types should be displayed in constructor and method signatures.

Group **diagrams.type.diagram.insets**

- Property package accepts non-negative integer values.
  This property specifies the distance from a package node's border to the type nodes contained in the package node.

Group **diagrams.type.diagram.order**

- Property fields accepts one of the ordering enumeration values.
  This property specifies the order of fields in type diagrams.
- Property constructors accepts one of the ordering enumeration values.
  This property specifies the order of constructors in type diagrams.
- Property methods accepts one of the ordering enumeration values.
  This property specifies the order of methods in type diagrams.

The following ordering enumeration values are available:

- lex
  Members are sorted according to their qualified names (and signatures in case of constructors and methods).
- lex-ic

Same as *lex*, but case insensitive.

- mod-lex
  Members are sorted according to their modifiers. Modifiers are considered to imply the following order:
  static public < public < static protected < protected < static package-private < package-private < static private < private
  If two (or more) members are equal according to this ordering, they are sorted according to their qualified names (and signatures in case of constructors and methods).
- mod-lex-ic
  Same as *mod-lex*, but case insensitive.

Group **diagrams.type.diagram.layout**

- Property PACKAGE_DISTANCE accepts values non-negative decimal values.
  This property specifies the distance between adjacent package nodes.
- Property RELATION_BUS_ROUTING accepts values true and false.
  This property specifies whether multiple generalization or realization edges should be routed in a bus-style manner.
- Property RELATION_DISTANCE accepts values non-negative decimal values.
  This property specifies the distance between the detailed type node and related type nodes.
- Property RELATION_TYPE_ALIGNMENT accepts one of the following values:

  - LEFT
  - CENTER
  - RIGHT
  - SHORTEST_DISTANCE
    Association nodes will be right aligned, dependency nodes left aligned.
  - LONGEST_DISTANCE
    Association nodes will be left aligned, dependency nodes right aligned.

  This property specifies specifies the alignment policy for association and dependency type nodes. If package nodes are displayed, alignment calculation is done on a per relation package basis.
- Property RELATION_TYPE_DISTANCE accepts values non-negative decimal values.
  This property specifies the distance between adjacent relation type nodes.
- Property RELATION_LABEL_LAYOUT_POLICY accepts one of the following values:

  - AS_IS
    No new label position is calculated.
  - OUTWARDS

For labels outside of a type node a new label position is calculated. For labels belonging to generalization or realization types, the new position is above the node, for labels belonging to association types the new position is to the left of the node, and for labels belonging to dependency types, it is to the right of the node.

This property specifies the label layout policy for labels of relation type nodes.

Group **formats**

Settings related to file formats of the generated UML diagrams and the way they are embedded into the HTML API documentation.

- Property fileformat accepts one of the following values:
    - GIF
      Well-known image format.
    - JPG
      Well-known image format.
    - PNG
      Well-known image format, the default.
    - SVG
      Scalable Vector Graphics, a XML-based vector graphics format.
    - SVGZ
      Compressed *SVG*.
    - SWF
      Shockwave Flash, a popular binary vector graphics format.

  This property specifies the file format for the generated UML diagrams.

Group **formats.vectorgraphics.display**

- Property scaling accepts one of the following values:
    - FIXED_SIZE
      The diagram will be displayed in a fixed size canvas (specified by properties width and height).
    - ACTUAL_SIZE
      The diagram will be displayed in a canvas sized to the diagram's actual size.
      This mode ignores properties width and height.
    - ACTUAL_SIZE_MAX_WIDTH
      The diagram will be displayed in a canvas sized to the diagram's actual size up to a fixed canvas width specified by property width).
      This mode ignores property height.
    - ACTUAL_SIZE_MAX_HEIGHT
      The diagram will be displayed in a canvas sized to the

    diagram's actual size up to a fixed canvas height (specified by property height).
    This mode ignores property width.
  - ACTUAL_SIZE_MAX_WIDTH_MAX_HEIGHT
    The diagram will be displayed in a canvas sized to the diagram's actual size up to a fixed canvas size (specified by properties width and height).
  - FIT_TO_SIZE
    The diagram will be scaled to fit into a canvas with fixed width and fixed height (specified by properties width and height).
  - FIT_TO_SIZE_BY_WIDTH
    The diagram will be scaled to fit into a canvas with fixed width (specified by property width).
    This mode ignores property height
  - FIT_TO_SIZE_BY_HEIGHT
    The diagram will be scaled to fit into a canvas with fixed height (specified by property height).
    This mode ignores property width.
  - SHRINK_TO_SIZE
    The diagram will be scaled to fit into a canvas with fixed width and fixed height (specified by properties width and height), unless it already fits.
  - SHRINK_TO_SIZE_BY_WIDTH
    The diagram will be scaled to fit into a canvas with fixed width (specified by property width), unless it already fits.
    This mode ignores property height.
  - SHRINK_TO_SIZE_BY_HEIGHT
    The diagram will be scaled to fit into a canvas with fixed height (specified by property height), unless it already fits.
    This mode ignores width.

  This property specifies the display scaling policy for UML diagrams. All policies will retain the diagram's original aspect ratio.
- Property width accepts positive integer values.
  This property specifies the canvas width for the generated UML diagram.
- Property height accepts positive integer values.
  This property specifies the canvas height for the generated UML diagram.
- Property reserveMinimum accepts values true and false.
  This property specifies whether yDoc should reserve a canvas at least the size of width and height when embedding UML diagrams into HTML API documentation.

Group **formats.vectorgraphics.svg**

- Property workaround accepts values true and false.
  This property specifies whether yDoc should use alternative HTML code for SVG embedding.

The alternative approach wraps <EMBED> tag(s) in <IFRAME> tags instead of <OBJECT> tags.

Group **formats.image**

- Property quality accepts values ranging from 0.0 to 1.0.
  This property specifies the compression quality for image formats that support compression (e.g. PNG, JPG).
  A compression quality setting of 0.0 is most generically interpreted as *high compression is important*, while a setting of 1.0 is most generically interpreted as *high image quality is important*.
- Property antialiasing accepts values true and false.
  This property specifies whether anti aliasing should be used in UML diagram image files.
- Property progressive accepts values true and false.
  This property specifies whether UML diagram image files should be encoded in progressive mode for image formats that support progressive encoding.
  Progressive encoding will result in image streams containing a series of scans of increasing quality.

Group **formats.image.tiling**

- Property enabled accepts values true and false.
  This property specifies whether UML diagrams should be written to multiple small image files instead of a single large one, if the image's width or height exceeds the corresponding maximum.
- Property width accepts non-negative integer values.
  This property specifies the maximum width for UML diagram image tiles.
- Property height accepts non-negative integer values.
  This property specifies the maximum height for UML diagram image tiles.

Group **misc**

- Property warnings accepts values true and false.
  This property specifies whether yDoc should emit warnings each time an explicit link (i.e. the result of @see or @link) to a documentation member, which was not accepted by the registered filters, is suppressed.

Group **misc.gc**

- Property frequency accepts non-negative integer values.
  This property specifies the number of UML diagrams to be generated between explicit calls to the Java garbage collector. A value of 10, for example, would result in a call to the garbage collector after every tenth diagram, whereas a

value of 1 will call garbage collection after each diagram.
A value of 0 will prevent yDoc from explicitly calling the Java garbage collector.

UML Styles

A style set defines the visual features of UML diagrams and is specified in a XML file which conforms to the yDoc style definition schema.
The yDoc distribution comes with several predefined style files, see **`resources/styles`**.
You can customize colors (main, border, text), fonts, shapes, and lines by either modifying an existing style file or creating a new one.
The yDoc 3.1 distribution includes StyleEd, a GUI-based style editor, that greatly simplifies customization. The editor is written completely in Java and will run on any Java 1.8 (or higher) runtime environment. All files needed to run StyleEd are contained in the executable JAR file **`lib/styleed.jar`**, i.e. double-clicking the file or invoking

```
java -jar <YID>/lib/styleed.jar
```

will start StyleEd.
Aside from style file editing, StyleEd also allows users to experiment with layout settings for the various diagram types.

7. Limitations

- The yDoc Evaluation version will only document ten classes.
  If one or more of those are excluded from documentation via the @y.exclude tag, they still count against that limit.
- In UML class diagrams generated by the yDoc Evaluation version, the associations list and the dependencies list will only display the ten above mentioned classes.

8. Acknowledgments

This product includes software developed by the Apache Software Foundation (http://www.apache.org/).
yDoc uses Batik to generate SVG files. Batik is distributed under the Apache Software License, Version 2.0.

yDoc uses FreeHEP VectorGraphics to generate SWF files. The FreeHEP VectorGraphics class library is distributed under the GNU Lesser General Public License.

Apache Software License

```
                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      (except as stated in this section) patent license to make, have made,
      use, offer to sell, sell, import, and otherwise transfer the Work,
      where such license applies only to those patent claims licensable
      by such Contributor that are necessarily infringed by their
      Contribution(s) alone or by combination of their Contribution(s)
      with the Work to which such Contribution(s) was submitted. If You
      institute patent litigation against any entity (including a
      cross-claim or counterclaim in a lawsuit) alleging that the Work
      or a Contribution incorporated within the Work constitutes direct
      or contributory patent infringement, then any patent licenses
      granted to You under this License for that Work shall terminate
      as of the date such litigation is filed.

   4. Redistribution. You may reproduce and distribute copies of the
      Work or Derivative Works thereof in any medium, with or without
      modifications, and in Source or Object form, provided that You
      meet the following conditions:

      (a) You must give any other recipients of the Work or
          Derivative Works a copy of this License; and

      (b) You must cause any modified files to carry prominent notices
          stating that You changed the files; and

      (c) You must retain, in the Source form of any Derivative Works
          that You distribute, all copyright, patent, trademark, and
          attribution notices from the Source form of the Work,
          excluding those notices that do not pertain to any part of
          the Derivative Works; and

      (d) If the Work includes a "NOTICE" text file as part of its
          distribution, then any Derivative Works that You distribute must
          include a readable copy of the attribution notices contained
          within such NOTICE file, excluding those notices that do not
          pertain to any part of the Derivative Works, in at least one
          of the following places: within a NOTICE text file distributed
          as part of the Derivative Works; within the Source form or
          documentation, if provided along with the Derivative Works; or,
          within a display generated by the Derivative Works, if and
          wherever such third-party notices normally appear. The contents
          of the NOTICE file are for informational purposes only and
          do not modify the License. You may add Your own attribution
          notices within Derivative Works that You distribute, alongside
          or as an addendum to the NOTICE text from the Work, provided
          that such additional attribution notices cannot be construed
          as modifying the License.

      You may add Your own copyright statement to Your modifications and
      may provide additional or different license terms and conditions
      for use, reproduction, or distribution of Your modifications, or
      for any such Derivative Works as a whole, provided Your use,
      reproduction, and distribution of the Work otherwise complies with
```

GNU Lesser General Public License

covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein.