RFC 208 Metatype Annotations

Final

29 Pages

Abstract

This RFC introduces annotations for the Metatype specification which can be use to annotate Java types so that tools can generate Meta Type Resources from the type declaration.

0 Document Information

0.1 License

DISTRIBUTION AND FEEDBACK LICENSE, Version 2.0

The OSGi Alliance hereby grants you a limited copyright license to copy and display this document (the “Distribution”) in any medium without fee or royalty. This Distribution license is exclusively for the purpose of reviewing and providing feedback to the OSGi Alliance. You agree not to modify the Distribution in any way and further agree to not participate in any way in the making of derivative works thereof, other than as a necessary result of reviewing and providing feedback to the Distribution. You also agree to cause this notice, along with the accompanying consent, to be included on all copies (or portions thereof) of the Distribution. The OSGi Alliance also grants you a perpetual, non-exclusive, worldwide, fully paid-up, royalty free, limited license (without the right to sublicense) under any applicable copyrights, to create and/or distribute an implementation of the Distribution that: (i) fully implements the Distribution including all its required interfaces and functionality; (ii) does not modify, subset, superset or otherwise extend the OSGi Name Space, or include any public or protected packages, classes, Java interfaces, fields or methods within the OSGi Name Space other than those required and authorized by the Distribution. An implementation that does not satisfy limitations (i)-(ii) is not considered an implementation of the Distribution, does not receive the benefits of this license, and must not be described as an implementation of the Distribution. "OSGi Name Space" shall mean the public class or interface declarations whose names begin with "org.osgi" or any recognized successors or replacements thereof. The OSGi Alliance expressly reserves all rights not granted pursuant to these limited copyright licenses including termination of the license at will at any time.

EXCEPT FOR THE LIMITED COPYRIGHT LICENSES GRANTED ABOVE, THE OSGi ALLIANCE DOES NOT GRANT, EITHER EXPRESSLY OR IMPLIEDLY, A LICENSE TO ANY INTELLECTUAL PROPERTY IT, OR ANY THIRD PARTIES, OWN OR CONTROL. Title to the copyright in the Distribution will at all times remain with the OSGi Alliance. The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted therein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

THE DISTRIBUTION IS PROVIDED "AS IS," AND THE OSGi ALLIANCE (INCLUDING ANY THIRD PARTIES THAT HAVE CONTRIBUTED TO THE DISTRIBUTION) MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DISTRIBUTION ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.
NEITHER THE OSGi ALLIANCE NOR ANY THIRD PARTY WILL BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE DISTRIBUTION.

Implementation of certain elements of this Distribution may be subject to third party intellectual property rights, including without limitation, patent rights (such a third party may or may not be a member of the OSGi Alliance). The OSGi Alliance is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

The Distribution is a draft. As a result, the final product may change substantially by the time of final publication, and you are cautioned against relying on the content of this Distribution. You are encouraged to update any implementation of the Distribution if and when such Distribution becomes a final specification.

The OSGi Alliance is willing to receive input, suggestions and other feedback (“Feedback”) on the Distribution. By providing such Feedback to the OSGi Alliance, you grant to the OSGi Alliance and all its Members a non-exclusive, non-transferable,

worldwide, perpetual, irrevocable, royalty-free copyright license to copy, publish, license, modify, sublicense or otherwise distribute and exploit your Feedback for any purpose. Likewise, if incorporation of your Feedback would cause an implementation of the Distribution, including as it may be modified, amended, or published at any point in the future ("Future Specification"), to necessarily infringe a patent or patent application that you own or control, you hereby commit to grant to all implementers of such Distribution or Future Specification an irrevocable, worldwide, sublicenseable, royalty free license under such patent or patent application to make, have made, use, sell, offer for sale, import and export products or services that implement such Distribution or Future Specification. You warrant that (a) to the best of your knowledge you have the right to provide this Feedback, and if you are providing Feedback on behalf of a company, you have the rights to provide Feedback on behalf of your company; (b) the Feedback is not confidential to you and does not violate the copyright or trade secret interests of another; and (c) to the best of your knowledge, use of the Feedback would not cause an implementation of the Distribution or a Future Specification to necessarily infringe any third-party patent or patent application known to you. You also acknowledge that the OSGi Alliance is not required to incorporate your Feedback into any version of the Distribution or a Future Specification.

I HEREBY ACKNOWLEDGE AND AGREE TO THE TERMS AND CONDITIONS DELINEATED ABOVE.

0.2 Trademarks

OSGi™ is a trademark, registered trademark, or service mark of the OSGi Alliance in the US and other countries. Java is a trademark, registered trademark, or service mark of Oracle Corporation in the US and other countries. All other trademarks, registered trademarks, or service marks used in this document are the property of their respective owners and are hereby recognized.

0.3 Feedback

This document can be downloaded from the OSGi Alliance design repository at https://github.com/osgi/design
The public can provide feedback about this document by opening a bug at https://www.osgi.org/bugzilla/.

0.4 Table of Contents

0.5 Terminology and Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in 1.

`Source code is shown in this typeface.`

0.6 Revision History

The last named individual in this history is currently responsible for this document.

Revision	Date	Comments
Initial	2013-11-19	Initial draft. BJ Hargrave, IBM
2nd	2013-11-20	Updated after feedback from Peter Kriens. Replaced Designate annotation with designate and designateFactory elements on the ObjectClassDefinition annotation. Also added icon element (and Icon annotation) to ObjectClassDefinition. BJ Hargrave. IBM
3rd	2013-11-21	Updates from CPEG meeting. Remove id element from AttributeDefinition. Change annotations to CLASS retention. Rename designate elements to pid. Allow negative cardinality values to mean List. Allow ObjectClassDefinition to be applied to interfaces. Update Meta Type spec to replace use of Vector with List. BJ Hargrave, IBM
4th	2013-12-05	Accepted changes after review at CPEG meeting. BJ Hargrave, IBM

Revision	Date	Comments
5th	2014-06-25	Added Designate annotation to replace pid/factoryPid elements of ObjectClassDefinition. BJ Hargrave, IBM
6th	2014-06-28	Restored replace pid/factoryPid elements of ObjectClassDefinition. BJ Hargrave, IBM
7th	2014-09-26	Updated from comments in bug 2725. BJ Hargrave, IBM
8th	2014-09-29	Clarify that ObjectClassDefinition annotation can be applied to annotation types and interface types. Use on concrete and abstract class types is not supported. BJ Hargrave, IBM
9th	2014-10-16	Update the descriptions to better handle interface types which can have much more varied methods than annotation types.. BJ Hargrave, IBM
10th	2014-10-17	Support option values and labels from List<Enum>, etc, BJ Hargrave, IBM
11th	2014-11-19	OCD.name and AD.name default values are based upon the id or method name, respectively, but the rules for creating the default values are not specified to allow freedom for the annotation processing tool. Clarified the supertypes of interfaces are also processed for ADs. BJ Hargrave, IBM
Final	2015-01-30	Mark Final for EG voting.

1 Introduction

The Metatype specification defines a Meta Type Resource format which can be used by Meta Type Service implementations. These resources are XML documents which conform to the Meta Type Resource XML Schema. RFC 190 introduces annotation configuration types to DS so that developers can access their configuration (component properties) in a type safe way. Since the configuration is now describable as a Java type, this RFC will also allow the type to document the Meta Type information about the configuration so that tools can generate Meta Type Resources from the Java type.

2 Application Domain

OSGi has long had the Meta Type specification which defines meta type information for configurations which are stored in Configuration Admin service. The Meta Type definitions are useful by GUIs to allow users to define actual configurations by providing information about the expected data types and values including localized information for a GUI. Meta Type specification also defines a Meta Type Resource format which is an XML document that can be contained in a bundle and processed by the Meta Type service.

Declarative Services uses configurations from Configuration Admin service as component properties for components. RFC 190 is updating DS to allow the component properties to be “shaped” into annotation types to provide components type-safe access to their component properties.

RFC 179 “DS Updates for Configurable” is an RFC which is no longer being worked but which contains the seed of the design now being using in RFC 190 for the configuration annotation types. RFC 179 is based upon RFC 178 “Configurable” which includes design ideas on annotations of these types for Meta Type support.

Bnd has also provided support for Meta Type annotations. See http://www.aqute.biz/Bnd/MetaType. The Meta.OCD and Meta.AD annotations were inputs to RFC 178.

3 Problem Description

Writing Meta Type Resource documents requires the programmer to author an XML document which both conforms to the Meta Type XML schema and accurately reflects the data and data types in the configuration. The programmer must keep changes to the program using the configuration and the XML document in sync. This can be difficult during refactoring and hard to validate during testing to avoid allowing errors from being propagated.

4 Requirements

MTA-0100 – Meta Type resource information must be able to be described in Java source code. This allows for compiler checking of types and refactoring support.

MTA-0200 – Must be able to mark a configuration annotation type (from RFC 190) as a source for Meta Type information.

MTA-0300 – Defaults for meta type information must be derivable from the marked type.

MTA-0400 – The programmer must be able to supply meta type information to override the defaults.

MTA-0500 – Tools must be able to process the meta type information specified in the source so that Meta Type Resource XML documents can be automatically generated.

MTA-0600 – Meta type information from the source files must also be present in the generated class files so tools do not need to process the source files.

5 Technical Solution

5.1 Introduction

Annotations are defined that can applied to the configuration annotation types from RFC 190 as well as interface types. An example from RFC 190:

```
@interface Config {
    boolean enabled() default true;
    String[] names() default {"a", "b"};
    String topic() default MyComponent.DEFAULT_TOPIC_PREFIX + "/topic";
}

@Component
public class MyComponent {
    static final String DEFAULT_TOPIC_PREFIX = "topic.prefix";
    protected void activate(Config configuration) {
        String t = configuration.topic();
    }
}
```

In this example, the Config annotation type is used as a configuration type which is used by the activate method. The Config type describes the “shape” of the configuration and can be used to also describe the meta type information. If we annotate the Config type with the new ObjectClassDefinition annotation,

```
@ObjectClassDefinition
@interface Config {
    boolean enabled() default true;
    String[] names() default {"a", "b"};
    String topic() default MyComponent.DEFAULT_TOPIC_PREFIX + "/topic";
}

@Component
@Designate(ocd = Config.class)
```

```
public class MyComponent {
    static final String DEFAULT_TOPIC_PREFIX = "topic.prefix";
    protected void activate(Config configuration) {
        String t = configuration.topic();
    }
}
```

a tool (like bnd) processing the bundle can automatically generate a Meta Type Resource XML document from the information in the Config type. The main purpose of these annotations is to generate Meta Type Resource XML documents from the configuration annotations. Tools processing these annotations must always generate valid Meta Type Resource XML documents. If this is not possible, then the tool must report an error to enable the programmer to take corrective action.

In this larger example:

```
@ObjectClassDefinition(localization = "OSGI-INF/l10n/test",
                       description = "%test.description",
                       name = "%test.name"
                       icon = @Icon(resource = "icon/test-32.png", size = 32))
public @interface Test {
            @AttributeDefinition(type = AttributeType.PASSWORD,
                       description = "%test.password.description",
                       name = "%test.password.name")
            public String _password();

            @AttributeDefinition(options = {
                       @Option(label = "%strategic", value = "strategic"),
                       @Option(label = "%principal", value = "principal"),
                       @Option(label = "%contributing", value = "contributing")
            },
                       defaultValue = "contributing",
                       description = "%test.membertype.description",
                       name = "%test.membertype.name")
            public String memberType();

            public String my_prop();
}

@Component(configurationPid = "test.pid")
@Designate(ocd = Config.class)
public class MyComponent {
    static final String DEFAULT_TOPIC_PREFIX = "topic.prefix";
    protected void activate(Config configuration) {
        String t = configuration.topic();
    }
}
```

we can see more extensive use of the new annotations. ObjectClassDefinition marks the Test type as a configuration type for which a meta type resource should be generated. It further defines meta type information including the description and name which are to be localized using the specified resource as well as an icon resource. AttributeDefinition marks elements of the Test type to provide meta type information. If meta type information is not provided by the annotation declaration, default information must be generated from the annotated type. The Designate annotation on the components connects the ObjectClassDefinition defined above with the pid of the component.

This RFC is tied to RFC 190 in that the annotations defined here are to be applied to the configuration annotation types defined by RFC 190. The annotations defined here can also be applied to interface types. However, the annotations defined here are unsupported on concrete and abstract class types.

5.2 @ObjectClassDefinition

The ObjectClassDefinition annotation is applied to a type to mark it for processing into a Meta Type Resource XML document. The ObjectClassDefinition annotation can be applied to configuration annotation types from RFC 190 as well as interface types. It is an error to apply it to an interface if any of the methods of the interface take arguments. It is an error to apply it to concrete or abstract class types. When applied to an interface, all the methods inherited from supertypes are include as AttributeDefinitions. The tool processing the annotations must be able to examine all the types in the hierarchy of the annotated interface to generate the Meta Type Resource. It is an error if the tool cannot examine a type in the hierarchy.

The ObjectClassDefinition annotation can be applied without defining any element values as defaults can be generated from the annotated type. The following elements are defined:

- name – (String) A human readable name of the object, can be localized if it starts with a % sign. The default is a string derived from the id. For example, _ and $ are replaced with space and space is inserted between camel case words. The name becomes the value of the name attribute of the OCD element in the generated Meta Type Resource XML document.

- id – (String) The id of the object, the default is the fully qualified name of the type with a $ as separator for nested classes. This is not to be confused with a PID which can be specified by the pid or factoryPid element. The id becomes the value of the id attribute of the OCD element in the generated Meta Type Resource XML document.

- localization – (String) The localization resource of the object. This refers to a resource property entry in the bundle that can be augmented with locale information. The default is the fully qualified name of the class prefixed by “OSGI-INF/l10n/”. The localization becomes the value of the localization attribute of the OCD element in the generated Meta Type Resource XML document.

- description – (String) A human readable description that can be localized when it starts with %. Default is the empty string. The description becomes the value of the description attribute of the OCD element in the generated Meta Type Resource XML document.

- pid – (String[]) The PIDs associated with the ObjectClassDefinition. The default is no associated PIDs. The pid information becomes a set of Designate elements for each pid which reference the OCD element in the generated Meta Type Resource XML document.

- factoryPid – (String[]) The factory PIDs associated with the ObjectClassDefinition. The default is no associated factory PIDs. The factory pid information becomes a set of Designate elements for each factoryPid which reference the OCD element in the generated Meta Type Resource XML document.

- icon – (Icon[]) Specify icons (resource name and size). The default is no icon information. The icon information becomes a set of Icon elements of the OCD element in the generated Meta Type Resource XML document.

Each method of the type annotated by ObjectClassDefinition is mapped to an AD child element of the OCD element in the generated Meta Type Resource XML document. The AttributeDefinition annotation only needs to be applied if values other than the defaults are desired.

5.3 @AttributeDefinition

The AttributeDefinition annotation is an optional annotation which can applied to the methods in a type annotated by ObjectClassDefinition. Each method of the type annotated by ObjectClassDefinition is mapped to an AD child element of the OCD element in the generated Meta Type Resource XML document. The AttributeDefinition annotation only needs to be applied if values other than the defaults are desired. The id of the AttributeDefinition is generated from the method name as specified in RFC 190 section 5.6.2 (e.g. removal of dollar sign and converting underscore to dot). The id becomes the value of the id attribute of the AD element in the generated Meta Type Resource XML document and is used as the name of the configuration property. The following elements are defined:

- name – (String) A human readable name of the attribute, can be localized if it starts with a % sign. The default is a string derived from the method name. For example, _ and $ are replaced with space and space is inserted between camel case words. The name becomes the value of the name attribute of the AD element in the generated Meta Type Resource XML document.

- description – (String) A human readable description that can be localized if it starts with %. Default is the empty string. The description becomes the value of the description attribute of the AD element in the generated Meta Type Resource XML document.

- type – (AttributeType) The type of the attribute. This must be one of the types defined in the Metatype specification. The default is derived from the return type of the method. Class and Enum types are mapped to String. Annotation types are not supported. A tool processing the annotation should declare an error during processing in this case. If the return type is `List`, `Set`, `Collection`, `Iterable` or some type which can be determined at annotation processing time to (1) be a subtype of `Collection` and (2) have a public no argument constructor, then the type is derived from the generic type. For example, a return type of `List<String>` will be mapped to String. A return type of a single dimensional array is supported and the type is the component type of the array. It is an error to have a multi dimensional array. Any unrecognized type is mapped to String. The type is used to select the value of the type attribute of the AD element in the generated Meta Type Resource XML document.

- cardinality - (int) The cardinality of the attribute. If the return type is an array, the default value is a large positive number. If the return type is `List`, `Set`, `Collection`, `Iterable` or some type which can be determined at annotation processing time to (1) be a subtype of `Collection` and (2) have a public no argument constructor, the default value is a large negative number. Otherwise, the default value is 0. The cardinality becomes the value of the cardinality attribute of the AD element in the generated Meta Type Resource XML document.

- min – (String) The minimum value allowed for this attribute. There is no default. The min becomes the value of the min attribute of the AD element in the generated Meta Type Resource XML document.

- max – (String) The maximum value allowed for this attribute. There is no default. The max becomes the value of the max attribute of the AD element in the generated Meta Type Resource XML document.

- defaultValue – (String[]) The default values. The defaultValues are concatenated into a comma delimited list to becomes the value of the default attribute of the AD element in the generated Meta Type Resource XML document. If not specified, if the annotated member is an annotation element that has a `default` value, then the value of this element is the `default` value of the annotated element. Otherwise, there is no default value.

- required – (boolean) Indicates if this attribute is required. The default is true. The required becomes the value of the required attribute of the AD element in the generated Meta Type Resource XML document.

- options - (Option[]) Specify options (value and optional label). There is only a default if the return type of the method is an Enum, Enum[], or a `List`, `Set`, `Collection`, `Iterable` or some type which can be determined at annotation processing time to (1) be a subtype of `Collection` and (2) have a public no argument constructor with a generic type of Enum in which case the label and the value is the enum value's name. The options information becomes a set of Option elements of the AD element in the generated Meta Type Resource XML document.

5.4 @Designate

The Designate annotation can be applied to a Declarative Services component class to make the connection between the pid of the component and an ObjectClassDefinition. This annotation must be used on a type that is also annotated with the Declarative Services @Component annotation. The component must only have a single PID which is used for the generated Designate} element.

- ocd – (Class) A class which is annotated with the ObjectClassDefinition annotation. The id of the referenced ObjectClassDefinition is used for the ocdref attribute in the generated Designate element.

- factory – (boolean) If false, then the PID value from the annotated component will be used in the attribute of the generated Designate element. If true, then the PID value from the annotated component will be used in the factoryPid attribute of the generated Designate element.

5.5 @Option

The Option annotation is only used for the options element of the AttributeDefinition annotation to allow specifying label/value pair for an AttributeDefinition.

5.6 @Icon

The Icon annotation is only used for the icon element of the ObjectClassDefinition annotation to allow specifying a icon resource/size pair.

5.7 Other Changes

Since this RFC will modify the Meta Type Specification and bump its version to 1.3, we can also pick up some minor Meta Type bugs awaiting a specification version change. The metatype package should also be updated to use the new package and type annotations from RFC 197.

5.7.1 Bug 2436

The schema is fixed to use “Character” instead of “Char” to match the proper Java type name and other OSGi specifications like DS and RSA.

5.7.2 Bug 2540

The schema is modified to allow more flexible ordering of elements.

5.7.3 List replaces Vector

The Meta Type specification will be updated to refer to List instead of Vector. A Vector is a List and List is the more modern type

6 Javadoc

OSGi Javadoc

11/19/14 2:10 PM

Package Summary		Page
org.osgi.service.metatype.annotations	Metatype Annotations Package Version 1.3.	14

Package org.osgi.service.metatype.annotations

```
@org.osgi.annotation.versioning.Version(value="1.3")
```

Metatype Annotations Package Version 1.3.

See:
Description

Enum Summary		Page
AttributeType	Attribute types for the `AttributeDefinition` annotation.	19

Annotation Types Summary		Page
AttributeDefinition	`AttributeDefinition` information for the annotated method.	15
Designate	Generate a `Designate` element in the Meta Type Resource for an `ObjectClassDefinition` using the annotated Declarative Services component.	23
Icon	`Icon` information for an `ObjectClassDefinition`.	24
ObjectClassDefinition	Generate a Meta Type Resource using the annotated type.	25
Option	`Option` information for an `AttributeDefinition`.	28

Package org.osgi.service.metatype.annotations Description

Metatype Annotations Package Version 1.3.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. This package has two types of users: the consumers that use the API in this package and the providers that implement the API in this package.

Example import for consumers using the API in this package:

```
Import-Package: org.osgi.service.metatype.annotations; version="[1.3,2.0)"
```

Example import for providers implementing the API in this package:

```
Import-Package: org.osgi.service.metatype.annotations; version="[1.3,1.4)"
```

Annotation Type AttributeDefinition

org.osgi.service.metatype.annotations

```
@Retention(value=RetentionPolicy.CLASS)
@Target(value=ElementType.METHOD)
public @interface AttributeDefinition
```

`AttributeDefinition` information for the annotated method.

Each method of a type annotated by `ObjectClassDefinition` has an implied AttributeDefinition annotation. This annotation is only used to specify non-default AttributeDefinition information.

The `id` of this AttributeDefinition is generated from the name of the annotated method. The annotated method name is processed from left to right changing each character as follows:

- A dollar sign (`'$'` \u0024) is removed unless it is followed by another dollar sign in which case the two consecutive dollar signs (`'$$'`) are changed to a single dollar sign.
- A low line (`'_'` \u005F) is changed to a full stop (`'.'` \u002E) unless is it followed by another low line in which case the two consecutive low lines (`'__'`) are changed to a single low line.
- All other characters are unchanged.

This id is the value of the id attribute of the generate AD element and is used as the name of the corresponding configuration property.

This annotation is not processed at runtime. It must be processed by tools and used to contribute to a Meta Type Resource document for the bundle.

See Also:
"The AD element of a Meta Type Resource."

Required Element Summary		Page
int	**cardinality** The cardinality of this AttributeDefinition.	16
String[]	**defaultValue** The default value for this AttributeDefinition.	17
String	**description** The human readable description of this AttributeDefinition.	16
String	**max** The maximum value for this AttributeDefinition.	17
String	**min** The minimum value for this AttributeDefinition.	17
String	**name** The human readable name of this AttributeDefinition.	15
Option[]	**options** The option information for this AttributeDefinition.	18
boolean	**required** The required value for this AttributeDefinition.	18
AttributeType	**type** The type of this AttributeDefinition.	16

Element Detail

name

```
public abstract String name
```

The human readable name of this AttributeDefinition.

If not specified, the name of this AttributeDefinition is derived from the name of the annotated method. For example, low line ('_' \u005F) and dollar sign ('$' \u0024) are replaced with space (' ' \u0020) and space is inserted between camel case words.

If the name begins with the percent sign ('%' \u0025), the name can be localized.

Default:
""
See Also:
"The name attribute of the AD element of a Meta Type Resource."

---

description

```
public abstract String description
```

The human readable description of this AttributeDefinition.

If not specified, the description of this AttributeDefinition is the empty string.

If the description begins with the percent sign ('%' \u0025), the description can be localized.

Default:
""
See Also:
"The description attribute of the AD element of a Meta Type Resource."

---

type

```
public abstract AttributeType type
```

The type of this AttributeDefinition.

This must be one of the defined attributes types.

If not specified, the type is derived from the return type of the annotated method. Return types of `Class` and `Enum` are mapped to STRING. If the return type is `List`, `Set`, `Collection`, `Iterable` or some type which can be determined at annotation processing time to

1. be a subtype of `Collection` and
2. have a public no argument constructor,

then the type is derived from the generic type. For example, a return type of `List<String>` will be mapped to STRING. A return type of a single dimensional array is supported and the type is the component type of the array. Multi dimensional arrays are not supported. Annotation return types are not supported. Any unrecognized type is mapped to STRING. A tool processing the annotation should declare an error for unsupported return types.

Default:
AttributeType.STRING
See Also:
"The type attribute of the AD element of a Meta Type Resource."

---

cardinality

```
public abstract int cardinality
```

The cardinality of this AttributeDefinition.

If not specified, the cardinality is derived from the return type of the annotated method. For an array return type, the cardinality is a large positive value. If the return type is `List`, `Set`, `Collection`, `Iterable` or some type which can be determined at annotation processing time to

1. be a subtype of `Collection` and
2. have a public no argument constructor,

the cardinality is a large negative value. Otherwise, the cardinality is 0.

Default:
0
See Also:
"The cardinality attribute of the AD element of a Meta Type Resource."

---

min

```
public abstract String min
```

The minimum value for this AttributeDefinition.

If not specified, there is no minimum value.

Default:
""
See Also:
"The min attribute of the AD element of a Meta Type Resource."

---

max

```
public abstract String max
```

The maximum value for this AttributeDefinition.

If not specified, there is no maximum value.

Default:
""
See Also:
"The max attribute of the AD element of a Meta Type Resource."

---

defaultValue

```
public abstract String[] defaultValue
```

The default value for this AttributeDefinition.

The specified values are concatenated into a comma delimited list to become the value of the `default` attribute of the generated `AD` element.

If not specified and the annotated method is an annotation element that has a `default` value, then the value of this element is the `default` value of the annotated element. Otherwise, there is no default value.

Default:
{}
See Also:
"The default attribute of the AD element of a Meta Type Resource."

---

required

```
public abstract boolean required
```

The required value for this AttributeDefinition.

If not specified, the value is `true`.

Default:
true
See Also:
"The required attribute of the AD element of a Meta Type Resource."

---

options

```
public abstract Option[] options
```

The option information for this AttributeDefinition.

For each specified `Option`, an `Option` element is generated for this AttributeDefinition.

If not specified, the option information is derived from the return type of the annotated method. If the return type is an `enum`, a single dimensional array of an `enum`, or a `List`, `Set`, `Collection`, `Iterable` or some type which can be determined at annotation processing time to

1. be a subtype of `Collection` and
2. have a public no argument constructor,

with a generic type of an `enum`, then the value of this element has an `Option` for each value of the `enum`. The label and value of each `Option` are set to the name of the corresponding `enum` value. Otherwise, no `Option` elements will be generated.

Default:
{}
See Also:
"The Option element of a Meta Type Resource."

Enum AttributeType

org.osgi.service.metatype.annotations

```
java.lang.Object
  └ java.lang.Enum<AttributeType>
      └ org.osgi.service.metatype.annotations.AttributeType
```

All Implemented Interfaces:
Comparable<AttributeType>, Serializable

```
public enum AttributeType
extends Enum<AttributeType>
```

Attribute types for the `AttributeDefinition` annotation.

See Also:
`AttributeDefinition.type()`

Enum Constant Summary	Page
BOOLEAN The `Boolean` type.	21
BYTE The `Byte` type.	20
CHARACTER The `Character` type.	20
DOUBLE The `Double` type.	21
FLOAT The `Float` type.	21
INTEGER The `Integer` type.	20
LONG The `Long` type.	20
PASSWORD The `Password` type.	21
SHORT The `Short` type.	20
STRING The `String` type.	20

Method Summary		Page
String	**toString**()	22
static AttributeType	**valueOf**(String name)	21
static AttributeType[]	**values**()	21

Enum Constant Detail

STRING

```
public static final AttributeType STRING
```

The `String` type.

Attributes of this type should be stored as `String`, `List<String>` or `String[]` objects, depending on the `cardinality` value.

LONG

```
public static final AttributeType LONG
```

The `Long` type.

Attributes of this type should be stored as `Long`, `List<Long>` or `long[]` objects, depending on the `AttributeDefinition#cardinality() cardinality` value.

INTEGER

```
public static final AttributeType INTEGER
```

The `Integer` type.

Attributes of this type should be stored as `Integer`, `List<Integer>` or `int[]` objects, depending on the `AttributeDefinition#cardinality() cardinality` value.

SHORT

```
public static final AttributeType SHORT
```

The `Short` type.

Attributes of this type should be stored as `Short`, `List<Short>` or `short[]` objects, depending on the `AttributeDefinition#cardinality() cardinality` value.

CHARACTER

```
public static final AttributeType CHARACTER
```

The `Character` type.

Attributes of this type should be stored as `Character`, `List<Character>` or `char[]` objects, depending on the `AttributeDefinition#cardinality() cardinality` value.

BYTE

```
public static final AttributeType BYTE
```

The `Byte` type.

Attributes of this type should be stored as `Byte`, `List<Byte>` or `byte[]` objects, depending on the `AttributeDefinition#cardinality() cardinality` value.

DOUBLE

`public static final AttributeType DOUBLE`

The `Double` type.

Attributes of this type should be stored as `Double`, `List<Double>` or `double[]` objects, depending on the `AttributeDefinition#cardinality() cardinality` value.

FLOAT

`public static final AttributeType FLOAT`

The `Float` type.

Attributes of this type should be stored as `Float`, `List<Float>` or `float[]` objects, depending on the `AttributeDefinition#cardinality() cardinality` value.

BOOLEAN

`public static final AttributeType BOOLEAN`

The `Boolean` type.

Attributes of this type should be stored as `Boolean`, `List<Boolean>` or `boolean[]` objects depending on `AttributeDefinition#cardinality() cardinality`.

PASSWORD

`public static final AttributeType PASSWORD`

The `Password` type.

Attributes of this type must be stored as `String`, `List<String>` or `String[]` objects depending on `cardinality`.

A `Password` must be treated as a `String` but the type can be used to disguise the information when displayed to a user to prevent it from being seen.

Method Detail

values

`public static AttributeType[] values()`

---

valueOf

`public static AttributeType valueOf(String name)`

---

toString

```
public String toString()
```

Overrides:
`toString` in class `Enum`

Annotation Type Designate

org.osgi.service.metatype.annotations

```
@Retention(value=RetentionPolicy.CLASS)
@Target(value=ElementType.TYPE)
public @interface Designate
```

Generate a `Designate` element in the Meta Type Resource for an `ObjectClassDefinition` using the annotated Declarative Services component.

This annotation must be used on a type that is also annotated with the Declarative Services `Component` annotation. The component must only have a single PID which is used for the generated `Designate` element.

This annotation is not processed at runtime. It must be processed by tools and used to contribute to a Meta Type Resource document for the bundle.

See Also:
"The Designate element of a Meta Type Resource."

Required Element Summary		Page
boolean	**factory** Specifies whether this Designate is for a factory PID.	23
Class<?>	**ocd** The type of the `ObjectClassDefinition` for this Designate.	23

Element Detail

ocd

```
public abstract Class<?> ocd
```

The type of the `ObjectClassDefinition` for this Designate.

The specified type must be annotated with `ObjectClassDefinition`.

See Also:
"The ocdref attribute of the Designate element of a Meta Type Resource."

---

factory

```
public abstract boolean factory
```

Specifies whether this Designate is for a factory PID.

If `false`, then the PID value from the annotated component will be used in the `pid` attribute of the generated `Designate` element. If `true`, then the PID value from the annotated component will be used in the `factoryPid` attribute of the generated `Designate` element.

Default:
false
See Also:
"The pid and factoryPid attributes of the Designate element of a Meta Type Resource."

Annotation Type Icon

org.osgi.service.metatype.annotations

```
@Retention(value=RetentionPolicy.CLASS)
@Target(value={})
public @interface Icon
```

`Icon` information for an `ObjectClassDefinition`.

See Also:
`ObjectClassDefinition.icon()`

Required Element Summary		Page
String	**resource** The resource name for this Icon.	24
int	**size** The pixel size of this Icon.	24

Element Detail

resource

```
public abstract String resource
```

The resource name for this Icon.

The resource is a URL. The resource URL can be relative to the root of the bundle containing the Meta Type Resource.

If the resource begins with the percent sign (`'%'` \u0025), the resource can be `localized`.

See Also:
"The resource attribute of the Icon element of a Meta Type Resource."

size

```
public abstract int size
```

The pixel size of this Icon.

For example, 32 represents a 32x32 icon.

See Also:
"The size attribute of the Icon element of a Meta Type Resource."

Annotation Type ObjectClassDefinition

org.osgi.service.metatype.annotations

```
@Retention(value=RetentionPolicy.CLASS)
@Target(value=ElementType.TYPE)
public @interface ObjectClassDefinition
```

Generate a Meta Type Resource using the annotated type.

This annotation can be used without defining any element values since defaults can be generated from the annotated type. Each method of the annotated type has an implied AttributeDefinition annotation if not explicitly annotated.

This annotation may only be used on annotation types and interface types. Use on concrete or abstract class types is unsupported. If applied to an interface then all methods inherited from super types are included as attributes.

This annotation is not processed at runtime. It must be processed by tools and used to generate a Meta Type Resource document for the bundle.

See Also:
"The OCD element of a Meta Type Resource."

Required Element Summary		Page
String	**description** The human readable description of this ObjectClassDefinition.	26
String[]	**factoryPid** The factory PIDs associated with this ObjectClassDefinition.	27
Icon[]	**icon** The icon resources associated with this ObjectClassDefinition.	27
String	**id** The id of this ObjectClassDefinition.	25
String	**localization** The localization resource of this ObjectClassDefinition.	26
String	**name** The human readable name of this ObjectClassDefinition.	26
String[]	**pid** The PIDs associated with this ObjectClassDefinition.	26

Element Detail

id

```
public abstract String id
```

The id of this ObjectClassDefinition.

If not specified, the id of this ObjectClassDefinition is the fully qualified name of the annotated type using the dollar sign ('$' \u0024) to separate nested class names from the name of their enclosing class. The id is not to be confused with a PID which can be specified by the pid() or factoryPid() element.

Default:
""

See Also:
"The id attribute of the OCD element of a Meta Type Resource."

name

```
public abstract String name
```

The human readable name of this ObjectClassDefinition.

If not specified, the name of this ObjectClassDefinition is derived from the id(). For example, low line ('_' \u005F) and dollar sign ('$' \u0024) are replaced with space (' ' \u0020) and space is inserted between camel case words.

If the name begins with the percent sign ('%' \u0025), the name can be localized.

Default:
""

See Also:
"The name attribute of the OCD element of a Meta Type Resource."

---

description

```
public abstract String description
```

The human readable description of this ObjectClassDefinition.

If not specified, the description of this ObjectClassDefinition is the empty string.

If the description begins with the percent sign ('%' \u0025), the description can be localized.

Default:
""

See Also:
"The description attribute of the OCD element of a Meta Type Resource."

---

localization

```
public abstract String localization
```

The localization resource of this ObjectClassDefinition.

This refers to a resource property entry in the bundle that can be augmented with locale information. If not specified, the localization resource for this ObjectClassDefinition is the string "OSGI-INF/l10n/" followed by the id().

Default:
""

See Also:
"The localization attribute of the MetaData element of a Meta Type Resource."

---

pid

```
public abstract String[] pid
```

The PIDs associated with this ObjectClassDefinition.

For each specified PID, a Designate element with a pid attribute is generated that references this ObjectClassDefinition.

The Designate annotation can also be used to associate a Declarative Services component with an ObjectClassDefinition and generate a Designate element.

Default:
{}
See Also:
"The pid attribute of the Designate element of a Meta Type Resource.", Designate

---

factoryPid

```
public abstract String[] factoryPid
```

The factory PIDs associated with this ObjectClassDefinition.

For each specified factory PID, a `Designate` element with a factoryPid attribute is generated that references this ObjectClassDefinition.

The Designate annotation can also be used to associate a Declarative Services component with an ObjectClassDefinition and generate a `Designate` element.

Default:
{}
See Also:
"The factoryPid attribute of the Designate element of a Meta Type Resource.", Designate

---

icon

```
public abstract Icon[] icon
```

The icon resources associated with this ObjectClassDefinition.

For each specified Icon, an `Icon` element is generated for this ObjectClassDefinition. If not specified, no `Icon` elements will be generated.

Default:
{}
See Also:
"The Icon element of a Meta Type Resource."

Annotation Type Option

org.osgi.service.metatype.annotations

```
@Retention(value=RetentionPolicy.CLASS)
@Target(value={})
public @interface Option
```

`Option` information for an `AttributeDefinition`.

See Also:
`AttributeDefinition.options()`

Required Element Summary		Page
String	**label** The human readable label of this Option.	28
String	**value** The value of this Option.	28

Element Detail

label

```
public abstract String label
```

The human readable label of this Option.

If not specified, the label of this Option is the empty string.

If the label begins with the percent sign (`'%'` \u0025), the label can be `localized`.

Default:
""

See Also:
"The label attribute of the Option element of a Meta Type Resource."

value

```
public abstract String value
```

The value of this Option.

See Also:
"The value attribute of the Option element of a Meta Type Resource."

7 Considered Alternatives

8 Security Considerations

The annotations do not have any security considerations.

9 Document Support

9.1 References

[1]. Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, RFC2119, March 1997.

[2]. Software Requirements & Specifications. Michael Jackson. ISBN 0-201-87712-0

9.2 Author's Address

Name	BJ Hargrave
Company	IBM

9.3 Acronyms and Abbreviations

9.4 End of Document