# Writing Javadoc

## Levels of support

We differentiate two levels of support for APIs:

- Supported API - The API is considered stable and there is a commitment to maintain compatibility between versions of the product. Changes made to supported APIs must be documented in the release notes.
- Evolving API - The API can be changed in minor/maintenance releases, there is no commitment to maintain binary compatibility from one version to another. Changes made to evolving APIs should be documented in the release notes.

## To mark an API supported

On latest version of AM this can be done by marking the relevant package/class/etc with the @Supported or @Evolving annotations. There are four annotation types available:

- `@Supported`
- `@SupportedAll`
- `@Evolving`
- `@EvolvingAll`

For versions prior to 7.0.0, there are four custom Javadoc tags that can be used:

- `@supported.all.api`
- `@supported.api`
- `@evolving.api`
- `@evolving.all.api`

The `@SupportedAll` and `@EvolvingAll` annotations can be used on classes or interfaces, and mark that every single component (field/constructor/method /class) of that class/interface should be part of the public Javadoc.

If you don't want to add a whole class to the public Javadoc, then add the `@Supported` or `@Evolving` annotation at the class level and use `@Supported` or `@Evolving` annotations to mark components one-by-one.

Any class that contains supported/evolving functionality **must** be marked public. Private and package-private classes are not included in the javadoc.

Note that methods and fields within these classes may still be generified when appropriate.

In case there is a new package involved for a new supported API, then:

- at most 1 package-info.java **HAS** to exist for the new package, and the Javadoc for the package **MUST** have a `@Supported` or `@Evolving` annotation.

When working on < AM 7.0.0 versions, use the relevant JavaDoc taglets as opposed to the annotations.

## Building the Javadoc

The product Javadoc can be built so you can see easily what is already supported API:

```
mvn clean source:jar javadoc:jar@attach-javadocs -pl openam-server-only -am -P\!xui,forgerock-release
```

The following might be an example of how to both build the Javadoc and then view it in your default browser:

```
mvn clean source:jar javadoc:jar@attach-javadocs -pl openam-server-only -am -P\!xui,forgerock-release
open openam-server-only/target/site/apidocs/index.html
```

The built Javadoc will also include all supported commons projects that might need to be used in order to implement an extension to AM - these are included automatically by including all modules that are dependencies of the openam-server-only module.

## When to add Supported API

The decision to include supported API on classes should be based on whether customers need the class in order to extend or subclass product code.

> *"As a rule of thumb, try implementing a plugin or extension to the system. If you find you need a product class to do this, then it should be supported API"*

When ever a supported class changes, we need to notify customers via a release notes change to make them aware that they might need to rebuild their custom plugins and related code.

Newly created APIs that are expected to be used in extensions should initially be marked as evolving, so that if necessary changes can be made as the new API stabilises in subsequent releases.

# How to change Supported API classes

If you find you need to change a supported API class, make the change as normal.

In addition, raise a JIRA ticket with the following fields:

- **Project**: AME
- **Issue Type**: Improvement
- **Component**: Documentation
- **Description**: Describe the impact on the customer, e.g. Will need to re-compile their code because of method signature change. Include necessary detail for deprecations by explaining what should be used instead.

An example of a ticket raised for this is AME-15439.