CDI Lite Bean Discovery

As you might know, there is an open discussion regarding handling bean discovery in CDI Lite. Basically, we are trying to come up with a solution in which:

- Lite bean archive will use a subset of what is in Full (e.g. you can transfer such archive to Full and it will behave equally regarding whether it is a bean archive and regarding discovery)
- Annotated will be a the go-to discovery in Lite as it is a natural fit
- It will allow defining that an archive discovery should be skipped (mode=none)

This document captures the options we have discussed over time (most or all of which are in one form or another in mailing list discussions) in order to help us make a decision.

Note: we have previously stated that in CDI Lite, we don't intend to include per-archive enablement of anything. Translated to beans.xml terms, the only thing that CDI Lite implementations would recognize is bean-discovery-mode; everything else is ignored. Enablement is only global (e.g. @Priority).

Link for sharing - https://docs.google.com/document/d/1WNTZpA4TLvunL5smfUhw_rboecRIJ9VfMwupxizjYyc/edit?usp=sharing

Option 1, (by Matej):

This option is based purely on what specification allows to this day and operates within its boundaries. We could declare bean discovery in Lite as follows:

- Lite requires beans.xml file to be present
- Lite requires this file to contain discovery mode and eligible modes are `annotated` and `none`
    - Any other mode would result in an error
- Any other content of beans.xml is NOT parsed
- Archives without beans.xml or with empty beans.xml result in non-portable behaviour

Pros:

- **No breaking change**, fits into the current specification

Cons:

- User hostile - users have to be aware of this and in order to create portable application, they are back to **mandatory presence of non-empty beans.xml**
- Still **contains cases which are non-portable** (empty/no beans.xml) and unless you can recompile such archive, you have no power over it
    - These cases may or may not be prevalent

Option 2, (by Matej):

Currently, when a CDI container detects empty beans.xml, it treats it as an explicit bean archive (in other words mode all). I suggest we change this so that beginning with the next major version, such archives will have discovery mode annotated. This of course makes it a breaking change but offers us a way to draft a solution that would then be portable between Full and Lite with no issues and would allow users to have minimal configuration.

Along with this, we'd also state in the specification that products are required to provide a switch that would allow users to return to previous behavior. This will be a safety latch giving users enough time to adjust their deployment. For the record, we already have a similar statement by the end of 12.1 Bean Archives.

With the above, the CDI Lite will then simply state that:

- You need to have beans.xml but it can be empty
- Annotated mode is default for empty beans.xml
    - Alternatively you can state discovery modes as annotated or none
- Nothing other than bean discovery mode is parsed

Notably, this option also puts forward annotated mode as a default in yet another aspect. This is something CDI has been promoting since annotated was introduced and it is IMO a good thing as annotated archives are what people really want in most cases because they are better performing, smaller and force people to consciously place annotations on their beans to determine their scopes and qualifiers which all what CDI is all about.

Pros:

- Simple user onboarding; users really just need an empty beans.xml which a lot of projects have anyway
- Annotated mode is now default without beans.xml as with empty beans.xml
- [Tomas] Aligns default discovery mode between Lite/SE/EE to annotated

Cons:

- Breaking change for existing application
    - The mandatory configuration option that allows switching to previous behavior solves this issue to certain extent

Option 3, (by Ladislav):

Another option is to add a new marker file, e.g. `META-INF/bean-archive`. The file must be empty, otherwise non-portable behavior results (this is for potential future extensibility).

CDI Lite implementations would treat JARs with this marker file as bean archives with annotated discovery, and they would treat JARs without this marker file as _not_ bean archives.

For CDI SE and CDI EE implementations, the following rules would apply:

- If a JAR includes `META-INF/bean-archive` and doesn't contain `META-INF/beans.xml`, then it is treated as if it contained `META-INF/beans.xml` with this content: `<beans bean-discovery-mode="annotated"/>`. Such JAR would be portable across all CDI implementations.
- If a JAR includes `META-INF/beans.xml`, then it's processed as usual (explicit bean archive). Such JAR would be portable across CDI SE and CDI EE implementations, but not necessarily across all implementations.
- If a JAR doesn't include either `META-INF/bean-archive` or `META-INF/beans.xml`, then it's processed as usual (implicit bean archive with annotated discovery). Such JAR would be portable across CDI SE and CDI EE implementations, but not necessarily across all implementations.

This leads to 2 cases that are portable across all CDI implementations:

- a JAR including `META-INF/bean-archive` and not including `META-INF/beans.xml`;
- a JAR not including `META-INF/bean-archive` and including `META-INF/beans.xml` with `bean-discovery-mode=”none”`.

All other cases are portable across CDI SE and CDI EE implementations, are also portable across CDI Lite implementations, but are not necessarily portable across all CDI implementations.

Main contribution is the idea of “portability across CDI SE and CDI EE implementations, but not necessarily across all implementations”. When carefully defined, this allows all existing applications (= using CDI SE or CDI EE) to function without change. At the same time, it lets us break out of the beans.xml jail, with limited (but existing!) options for compatibility across both worlds.

It would also be possible for CDI SE and CDI EE implementations to include a configuration switch to use the CDI Lite style of bean discovery (ignoring beans.xml, basically). That should default to off for compatibility reasons.

If we think there is a serious chance we might want to introduce some configuration in the new marker file, it could be called `bean-archive.properties` (or `beans.properties`).

Pros:

- Doesn’t break compatibility
- Relatively clean cut for Lite
- Doesn’t require XML parsing in Lite implementations

Cons:

- Relatively complex rules
    - That is because it provides more guarantees than e.g. option 1

    - The full complexity only concerns people that care about both Lite and SE/EE
    - For people who only care about Lite, this is simple: they can ignore META-INF/beans.xml
    - For people who only care about SE/EE, this is simple: they can ignore META-INF/bean-archive and keep doing what they've always been doing

Variant (or elaboration)

Say we go with `META-INF/beans.properties`. This is intuitively an analogue of `META-INF/beans.xml`, so probably a good choice.

We could already define one configuration property that could go into this file. That would be `bean-discovery-mode=none|annotated` (specifically excluding `all`, because we want to discourage that), or perhaps even `bean-archive=true|false` (`true` meaning "it's a bean archive with annotated discovery", `false` meaning "it's not a bean archive"; again, no option to define `all` discovery mode).

The file may be empty, in which case it is treated as if it contained `bean-archive=true`.

If the file contains any other configuration properties, non-portable behavior results.

For CDI Lite implementations, the following rules would apply:

- If a JAR contains `META-INF/beans.properties` with `bean-archive=true`, then it is a bean archive with annotated discovery.
- If a JAR doesn't contain `META-INF/beans.properties`, or if it contains `beans.properties` with `bean-archive=false`, then it is not a bean archive.
- `META-INF/beans.xml` is ignored.

For CDI SE and EE, the following rules would apply:

- If a JAR includes `META-INF/beans.properties` with `bean-archive=true` and doesn't contain `META-INF/beans.xml`, then it is treated as if it contained `META-INF/beans.xml` with this content: `<beans bean-discovery-mode="annotated"/>`. Such JAR would be portable across all CDI implementations.
- If a JAR includes `META-INF/beans.properties` with `bean-archive=false` and doesn't contain `META-INF/beans.xml`, then it is treated as if it contained `META-INF/beans.xml` with this content: `<beans bean-discovery-mode="none"/>`. Such JAR would be portable across all CDI implementations.
- If a JAR includes `META-INF/beans.xml`, then any potentially present `META-INF/beans.properties` is ignored and the JAR is processed as usual (explicit bean archive). Such JAR would be portable across CDI SE and EE implementations, but not necessarily across all implementations.
- If a JAR doesn't include either `META-INF/beans.properties` or `META-INF/beans.xml`, then it's processed as usual (implicit bean archive with annotated discovery). Such JAR would be portable across CDI SE and EE implementations, but not necessarily across all implementations.

As a result, there are more options to guarantee compatibility across all CDI implementations. A simple approach would be to ignore and never include `beans.xml`, and only use `beans.properties`.

Option 4, (by Antoine):

This option introduces a new optional cdi global configuration file : META-INF/cdi-config.xml. Its introduction goes beyond the bean discovery topic, since it could allow configuring other aspects of the app (interceptors, override priorities, etc…)

This file should be placed in the main application to override standard config defined in beans archive. If placed in a bean archive it should be ignored.
In this file we could configure the discovery mode for all bean archive or for a given one.
Possible (**non exhaustive**) scenarios could be :

- Force all bean archive with an empty beans.xml or without bean discovery mode to be in annotated mode
- Force all archive without a beans.xml to become a bean archive in annotated mode
- Switch a given bean archive to “none” discovery mode
- Add an annotation as a bean defined annotation
- Add a specific class as a bean

Pros:

- Stay compatible with current behavior : no breaking change
- Compatible with other options (it could be used to activate the option 2 explicitly)
- Move some Jakarta EE specific switches to a standard place

Cons:

- Yet another configuration file
- Could be confusing
- [Tomas] Hard to define how to discover this file in SE environment (as there is no war or ear file that would be “THE” application file)