Plugin file format - Variable expansion

Table of contents

1 Variable expansion...2
2 Global context...2
2.1 Variable env.name...2
3 Plugin context... 2
3.1 Variable plugin.id...3
3.2 Variable plugin.dir...3
3.3 Variable plugin.version... 3

1. Variable expansion

Much of the plugin file format allows variables to be expanded. This simplifies the process of defining plugins and allows paths, etc. not to be hard coded.

The format of expansions is relatively simple. A string is expanded by replacing each occurrence of `${`*`variable name`*`}` with the value of the variable. Thus, for example, when defining a shared library, the path to the library can be set as follows:

```
<library path="${plugin.dir}/library.so"/>
```

Variables are resolved in some context or other (the documentation for the attribute or element supporting expansion will indicate what the context is). There most common contexts is the plugin context.

The character `'$'` may be escaped by repeating it. So, if we have some variable, `foo`, with value, `bar`, then the following strings expand:

String	Expansion
The value of foo is ${foo}.	The value of foo is bar.
The value of $${foo} is ${foo}.	The value of ${foo} is bar.

2. Global context

The global context first looks for variables that are application defined. Applications can add as many global variables as they wish. Check the application documentation to see which variables are defined.

If the variable is still not found, then the context will check the folling variables in the order below, returning the first one found.

2.1. Variable env.name

This is a family of variables beginning with `env.`.

If the variable name starts with `env.` the context does then it will take the remainder of the name and try find an environment variable with that name, returning that.

3. Plugin context

The plugin context searches for variables in the following order.

First plugin defined variables are checked. These are created by having `variable` elements in the plugin specification (see the Element reference). For example, the following code creates two variables. The first is named `foo` with value `bar` and the second is called `mars` with value `chocolate bar`.

```
<variable name="foo" value="bar"/>
<variable name="mars" value="chocolate ${foo}"/>
```

Note:
The order of variable declaration is important. Had it been the other way around, then the value of `bar` would have been `chocolate` .

3.1. Variable plugin.id

This is the `id` attribute of the plugin.

3.2. Variable plugin.dir

This is the absolute directory path in which the plugin specification file is found.

3.3. Variable plugin.version

This is the `version` attribute of the plugin.

If the variable still has not been found, then the global context will be searched for the variable.

Note:
When an element indicates that some string is expanded in the context of the plugin, they mean in the context of plugin file in which the string is found. This means that if an extension point says that strings in extensions are expanded in the context of the plugin, it does not mean in its plugin context but in the context of the extending plugin.