Documentation about the codeRef

1. Introduction:

The goal of this document is to let the user know that there are different ways to specify the field <codeRef> in the XML file. This field appears in each component of the XML. Consider the following example:

```
<codeRef>lsim.LSimDispatcherHandler::http://localhost:8080/lsim-
frontend/experiments/files/dllamazares/testapp.zip$testapp.zip/coordinator::script.sh
</codeRef>
```

It is possible to distinguish 4 parts, everyone with a different color, and separated by special characters (“::” and “$”). What is the meaning of each part?

- `lsim.LSimDispatcherHandler`:

This part allways has the same value. It indicates the name of a class which implements the interface to communicate with the dispatcher, but it is no necessary to go into detail.

- `http://localhost:8080/lsim-frontend/experiments/files/dllamazares/testapp.zip`:

This part is the local path (in local environment), or the URL address (in distributed environment) where the dispatcher downloads the ZIP. If the ZIP was uploaded to the frontend, this part allways will follow this structure. However, this address could be any valid address, like a Dropbox public link.

- `testapp.zip/coordinator`:

This part indicates the ZIP directory where the component is located.

- `script.sh`:

This part indicates the name of the script that will be executed in order to start the component. By default, if it is not indicated, it is assumed that the dispatcher will use a generic script “script.sh” located in the dispatcher folder.

2. Using a local environment:

Once the <codeRef> structure is understood, now we can explain the different ways of specifying the field <codeRef>.

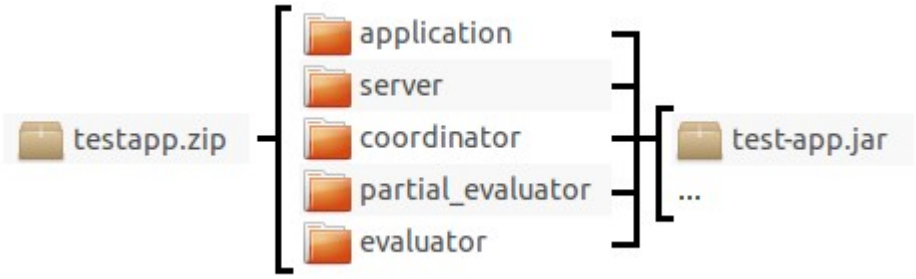
In this case, we have a local path instead of a URL. The codeRef can be written in two ways: Indicating an alternative script or not. If an alternative script is indicated, it must be located in the folder of the component.

```
<codeRef>lsim.LSimDispatcherHandler::localpath/coordinator</codeRef>

<codeRef>lsim.LSimDispatcherHandler::localpath/coordinator::scriptName.sh</codeRef>
```

3. Using a distributed environment:

3.1. Using a ZIP which contains a directory for each component:



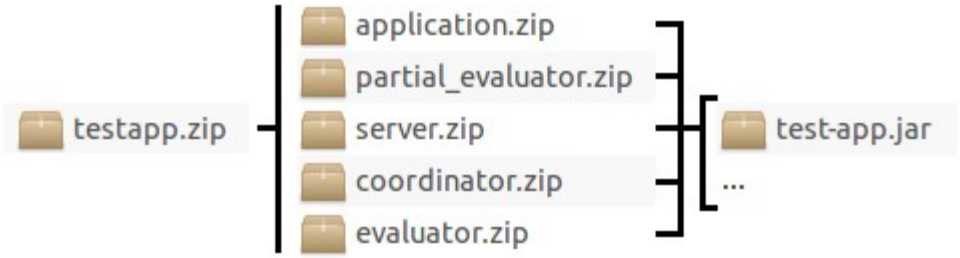
In this case, each component will download the whole ZIP. This is a bit inefficient because all components will download things that they don't need.

The codeRef of the XML can be written in two ways, but always with the same URL for each component because they download the same ZIP which contains everything. It is also needed to indicate always the directory of the ZIP. The flexible part in this case is the possibility of indicate an alternative script or not. If an alternative script is indicated, it must be located in the folder of the component.

```
<codeRef>lsim.LSimDispatcherHandler::URL/testapp.zip$testapp.zip/coordinator</codeRef>

<codeRef>lsim.LSimDispatcherHandler::URL/testapp.zip$testapp.zip/coordinator::scriptNa
me.sh</codeRef>
```

3.2. Using a different ZIP for each component:



In this case, each component will download their own ZIP. This is more efficient because all components will download only the things they need.

The codeRef of the XML can be written in two ways. All components have their own URL because they download their own ZIP. It is not necessary to indicate the ZIP directory where the component is located because it is the root of the ZIP. The flexible part in this case is the possibility of indicate an alternative script or not. If an alternative script is indicated, it must be located in the ZIP.

```
<codeRef>lsim.LSimDispatcherHandler::URL/coordinator.zip</codeRef>
```

```
<codeRef>lsim.LSimDispatcherHandler::URL/coordinator.zip::scriptName.sh</codeRef>
```

3. How does the frontend work?

In the first case (using a ZIP which contains a directory for each component), the frontend stores it in the temporal folder. Then, the zip can be referenced if the URL follows the right structure.

In the second case (using a zip which contains a zip for each component), the frontend stores it in the temporal folder and extracts here every component from it. Then, every zip component can be referenced if the URL follows the right structure.

Anyway, the URL of the codeRef, can point to any site in the Internet. It is to say, the URL can be any valid address, like a Dropbox public link. If all the URLs in the XML point to non-frontend addresses, of course it is not necessary to upload the ZIP to the frontend because the frontend only needs the XML file.