

xUML Service Model for Java Callback

As soon as the preconditions on Java side are met, you need to do the following things within your xUML service model to implement the Java callback:

1. Import the prepared JAR file into an xUML service as described on [Importing Java™ Classes and Properties Resource Files](#) in the E2E Builder User's Guide.
2. Define an `<<E2EJavaPort>>` implementing the offered callback interface (see [Implementing the Callback](#) further below).
3. Add a component diagram with a `<<JavaComponent>>` and an `<<E2EJavaService>>` (see [Java Callback Components](#) further below).

Implementing the Callback

The Java callback needs to be implemented by an `<<E2EJavaPort>>` on Bridge side.

<pre>classDiagram class JavaCallbackInterface { <<JavaCallbackOperation>> +sendText(text: String) } class HelloBridgePort { <<E2EJavaPort>> <<JavaCallbackOperation>> +sendText(string: String) } JavaCallbackInterface < -- HelloBridgePort</pre>	In the example, you can find a class HelloBridgePort with stereotype <code><<E2EJavaPort>></code> that implements the callback by a sendText operation. This implementing class needs to be linked to the <code><<JavaCallbackInterface>></code> by an Interface Realization relation. The implementing operation sendText itself needs to have stereotype <code><<JavaCallbackOperation>></code>.
	In the containment tree, you can see the port type and the operation.
	Operation sendText is implemented by an activity diagram, which uses parameter string to write a log entry.

On this Page:

- [Implementing the Callback](#)
- [Java Callback Components](#)

Related Pages:

- [Importing Java™ Classes and Properties Resource Files](#)
- [Java Implementation for Java Callback](#)

Java Callback Components

Having defined the implementation, you can already start building the component diagram using the wizard.

Figure: Java Callback Components

