Java Implementation for Java Callback

In order to invoke an activity diagram from a Java object, the Java implementation has to:

- implement the Java interface **BridgeJavaService** resp. **BridgeJavaServiceStartStopInterface**
- provide an interface to be implemented by the Bridge

The interface **BridgeJavaService** needs to be implemented within the Java application.

```
public class HelloWorldService implements
BridgeJavaService<HelloWorldCallback>, BridgeJavaServiceStartStopInterface
{

        public void initialize(final HelloWorldCallback callback) {
        // initialize service, should not access other components or may
cause race conditions
        }

        public void start() {
        // start serving requests. From this point on all components can
be freely accessed
        }

        public void shutDown() {
        // stop accepting new requests
        }

        public void stop() {
        // wait for active requests to finish and do some necessary cleanup
        }

 }
```

The interface contains the following methods:

Method	Description
initialize	Is invoked when the Bridge service is started. **initialize** should not access other components or this may cause race conditions.
start	Runtime 2017.4 Builder 6.0.22 Is invoked, when all service components are initialized.
shutdown	Runtime 2017.4 Builder 6.0.22 Are invoked, when all service components are shut down.
stop	

In the example, the Java class **HelloWorldJavaService** implements the interface **BridgeJavaService** and its operations.

Furthermore, within the Java application, an interface needs to be defined, which inherits from the generic interface **BridgeJavaCallback**. In the example, the Java Interface **HelloWorldCallback** is defined. This interface and its operations will later on be implemented by a UML class within the E2E BRIDGE, as defined in the following section xUML Service Model for Java Callback.

```
public interface HelloWorldCallback extends BridgeJavaCallback {

        public void sendText(String text);
 }
```

After considering the above preconditions in the Java code, you can create the JAR-file(s) which will be imported into the E2E Bridge. The import is described on Importing Java™ Classes and Properties Resource Files.

> The JAR-files for the illustrated example are located in the **jarfiles** directory of the E2E Builder project **Add-ons**. The JAR files also contain the Java sources, which show how to write importable classes.

On this Page:

Related Pages:

- xUML Service Model for Java Callback
- Importing Java™ Classes and Properties Resource Files

Migration Notes

Methods **start** and **stop** are new with Runtime 2017.4 and Builder 6.0.22. If you have implemented this class before, please note the following:

- If your class already contains start and/or stop methods but does not implement **BridgeJavaServiceStartStopInterface**, the methods **will not be executed**.
- **stop** needs Runtime 2017.4. Runtimes before 2017.4 will silently disregard **stop**, **start** will be executed though.