# Tuscany Runtime Launching

## Tuscany Runtime Launching

This page describes the different usage scenarios on Tuscany runtime is launched to run the SCA applications.

### User Type 1:

Wants to run a contribution/composite with not further coding and have a container that already embeds the Tuscany runtime to allow them to do this.

#### Supported mechanisms:

- Contribution Zip/Jar/Directory
  - Use container specific mechanisms for loading contribution
- Webapp (SCA)
  - construct a WAR and contribute to SCA enable container
- Enterprise (SCA)
  - construct an EAR and contribute to SCA enabled container

### User Type 2:

Wants to run a contribution/composite with not further coding. I.e. They don't want to write a mainline to start a node instead they want to treat Tuscany as a pre-compiled application. They will have downloaded a binary Tuscany distribution

#### Supported mechanisms:

- Tooling
  - Eclipse plugin, right click on composite file in contribution project. Relies on Eclipse Plugin
- Command line (includes running from Ant)
  - see * below, this relies on a TuscanyLauncher
- Webapp (Generic)
  - construct a WAR with appropriately configured web.xml using TuscanyServletFilter or [ TuscanyContextListener] as appropriate.

Command line. There are a number of variables we need to take account of here.

- OSGi/J2SE
- Standalone Node/Domain/Node registering with domain
- Normal/Debug (I just made this up)

Two obvious approaches

1. Different launcher classes to do different things.
2. One launcher that is parameterized based on what you want it to do.

### User Type 3

Wants to start the runtime from code as they want to embed it for some reason. Their reason could range from just wanting to automate testing right through to wanting to extend some existing software with SCA runtime capabilities. Will have downloaded a binary distribution. The pre-condition here is they have written some kind of mainline that uses Tuscany classes. It could fire up nodes using the NodeFactory or other parts Tuscany such as the model processing.

#### Supported mechanisms:

- Tooling
  - dependency on Tuscany library from Eclipse Plugin
  - dependency on Tuscany modules from distribution. Set up manually
  - dependency on Tuscany modules from local maven repo (mvn -Peclipse)
- Command line(includes running from Ant)
  - java MyClass.jar
  - Specify the required jars on the classpath either manually or with tuscany-sca-manifest.jar
- Mvn
  - include a dependency on appropriate Tuscany feature distributions/modules
- OSGi
  - Construct a bundle which uses some Tuscany classes. Add all the appropriate tuscany jars and dependencies to your OSGi environment. Start the application bundle.

### User Type 4

Wants to contribute to the Tuscany project or build extensions to Tuscany. Will have a source distribution. They may use various means to test infrastructure changes including the mechanisms used by User Types 1, 2 & 3.

## Supported mechanisms:

- Tooling
  - With mainline that uses either the launcher or other Tuscany classes include a dependency on the Tuscany loaded classes in the Eclipse workspace so any changes are picked up as they happen without a mvn compile. This can be achieved by including appropriate Tuscany modules from workspace (mvn -Peclipse). As we build Tuscany module as bundles using the PDE we rely on the PDE target distribution to configure the Eclipse evironment.
- Mvn
  - With mainline that uses either the launcher or other Tuscany classes include a dependency on Tuscany feature distributio/modules