**Installing Java Code Generator Plug-in under Eclipse**

1. Make sure, you have the Eclipse Plug-in Development Environment and Titan Designer Feature installed under Eclipse.
2. On the **Package Explorer** pane in Eclipse,right-click an empty spot and select **Import**.
3. In the pop-up window, under **General**, select **Existing Projects into Workspace**. Click Next.
4. Select the radio button named **Select root directory** then click **Browse** next to it and select the org.eclipse.titan.codegenerator directory, wherever you’ve downloaded it. Click **Finish**.
5. On the **Package Explorer** pane, click the arrow next to the newly imported org.eclipse.titan.codegenerator project to expand it.
6. Make sure that no other instances of the Code Generator plugin are installed in Eclipse (this is true if the coffee cup icon is not present on the toolbar). To uninstall existing instances of the plugin, go to Help/Installed Software (here, you should remove the Codegenerator plugin and its install patch as well). If it does not help, empty the eclipse/dropins folder. A search for files containing the text „codegenerator” in the workspace/.metadata folder can also help, as this way, you can get the paths on which, other instances of the plugin’s JAR are stored and then remove them by hand.
7. Go to org.eclipse.titan.codegenerator/src/org.eclipse.titan.codegenerator/walker.properties and set the variable javafile.path to the directory where you would like the Java files to be generated.
8. Double-click plugin.xml. Scroll to the bottom of the Overview tab, and click **Export Wizard** in the **Exporting** section.
9. In the pop-up window, make sure org.eclipse.titan.codegenerator is selected in the Available Plug-ins and Fragments section, then on the Destination select radio button **Install into host. Repository:**
10. Click **Finish**, and restart Eclipse when prompted to.

**Running TITAN projects**

1. Open your source TITAN project and open one of the TTCN-3 files in it.
2. Make sure you have a config file with .cfg extension in the src folder of the TITAN project. The config file should only contain the name of the test case to be executed, in the first row.
3. If you already have the necessary TP\_<port type name>\_PT.java test port files (the ones needed for non-internal TTCN-3 port types), make sure that these files are also in the src folder of the TITAN project.
4. You have 3 possibilities:
   1. If you’d like to cross-compile but not run your TITAN project, click on the coffee cup icon without the play sign („Generate JAVA code from the TTCN sources”). As a result, a new Java project is generated into the workspace under the name org.eclipse.titan.codegenerator.output. Its org.eclipse.titan.codegenerator.javagen package contains the files newly generated from the TTCN source, while its org.eclipse.titan.codegenerator.TTCN3JavaAPI package contains the files of the runtime API. Test port skeletons for all non-internal port types are also generated into the org.eclipse.titan.codegenerator.javagen package under the name TP\_<port type>\_PT.java. Starting off from these, test ports can be implemented. If the compiler finds test port files in the src folder of the TITAN project, the corresponding skeletons will be overwritten with these.
   2. If you’d like to cross-compile and run your TITAN project on multiple hosts, follow step a. Make sure that the completely implemented, necessary test ports are available in the src folder of the TITAN project in advance, or they are finalized after the cross-compilation, based on the skeletons, before proceeding. Make sure that the generated java project is present on each host. Then, follow the steps below:
      1. Set the number of hosts used for test execution in org.eclipse.titan.codegenerator.output/org.eclipse.titan.ttcn3java.TTCN3JavaAPI/MC.java, as the value of variable HCNUM.
      2. Set the IP address of the host where the main test component should be started, in the same file, in variable MTCIP.
      3. Set whether you’d like a detailed log or a less detailed one in variable DEBUGMODE, in the same class. You can set it for each host controller in the corresponding org.eclipse.titan.codegenerator.output/org.eclipse.titan. javagen/HC.java files as well.
      4. Set the output file of the logger in org.eclipse.titan.codegenerator.output /org.eclipse.titan.ttcn3java.TTCN3JavaAPI/TTCN3Logger.java, line 52.
      5. Right-click org.eclipse.titan.codegenerator.output/org.eclipse.titan.ttcn3java .TTCN3JavaAPI/MC.java, select Run As, then select Java Application.
      6. Right-click org.eclipse.titan.codegenerator.output /org.eclipse.titan.javagen /HC.java, select Run As, then select Java Application. Repeat this for each host that is used for test execution. After the last host controller is connected to the main controller, test execution starts automatically.
      7. If you’d like to set the parameters in steps i, ii, iii, and iv as default, do it under org.eclipse.titan.codegenerator/org.eclipse.titan.ttcn3java. TTCN3JavaAPI. In this case, it will not be necessary to re-set these parameters after each cross-compilation
   3. If you’d like to cross-compile and run your TITAN project right away, optionally change the parameters according to step b/vii, then click on the coffee cup icon with the play sign („Generate JAVA code from the TTCN sources and run”). Note that in this case, you must have all the necessary (completely implemented) test port files in the src folder of your TITAN project. Also note, that this feature is only available currently, when running the TITAN project on a single host. Of course, single-host execution can be performed according to step b, too.
5. To see the results of the test execution, either open the log file you specified, or see the results on the Eclipse console (the log file contains more entries even when not in debug mode).

**Note:** Please be aware that in the current state of the generator, when creating a test component, the create function has to be parameterized with the name of the test component and the IP address of the host on which it has to be created (so two charstring parameters in total). Please, also be aware that in the current state of the generator, the control parts of TTCN-3 modules are not processed. Instead, the name of the test case to be executed should be explicitely specified in the above mentioned config file, in the source project.