

User Guide of the TITAN Executor for the Eclipse IDE plug-in

Jenő Balaskó, Ádám Knapp

Version 11.2.1, 2026-08-05

Table of Contents

1. Introduction . . . 2
   1.1. Overview . . . 2
   1.2. Target Groups . . . 2
   1.3. Typographical Conventions . . . 2
   1.4. Installation . . . 2
   1.5. Reporting Errors . . . 2
2. Getting Started . . . 4
   2.1. The TITAN Executing Perspective . . . 4
3. Setting Workbench Preferences and Project Properties . . . 7
   3.1. TITAN Executor Preferences . . . 7
   3.2. TITAN Executor Project Properties . . . 8
4. Launching the Test Suite . . . 9
   4.1. The Launching Modes Supported by the TITAN Executor Plug-in . . . 9
   4.2. Creating Launch Configuration . . . 10
5. Executing and Controlling the Execution of Test Suites . . . 25
   5.1. Execution Control . . . 25
   5.2. TITAN Execution Controller View . . . 25
   5.3. TITAN Notifications View . . . 32
   5.4. TITAN Test Results View . . . 33
   5.5. Console Views . . . 34
   5.6. Limitations . . . 36
6. Other Available Functions . . . 37
   6.1. Formatting Log Files . . . 37
   6.2. Merging Log Files . . . 37
7. Launching TITAN Java Projects . . . 38
   7.1. The Launching Modes Supported by the TITAN Executor Plug-in for TITAN Java Projects . . . 38
8. References . . . 48

Abstract

This document describes detailed information of using the TITAN Executor for the Eclipse IDE plug-in.

Copyright

Copyright (c) 2000-2026 Ericsson Telecom AB.
All rights reserved. This program and the accompanying materials are made available under the terms of the Eclipse Public License v2.0 which accompanies this distribution, and is available at

https://www.eclipse.org/org/documents/epl-2.0/EPL-2.0.html.

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

Chapter 1. Introduction

1.1. Overview

This document describes the general workflow and use of the TITAN Executor for the Eclipse IDE plug-in.

TITAN Executor is a tool to launch testcases with different configuration types, and analyzing their results.

It is advised to have a basic knowledge and understanding of the Eclipse IDE and its workflows. For information, open **Help** > **Workbench User Guide** in the Eclipse GUI.

1.2. Target Groups

This document is intended for system administrators and users who intend to use the TITAN Executor plug-in for the Eclipse IDE.

1.3. Typographical Conventions

This document uses the following typographical conventions:

Bold is used to represent graphical user interface (GUI) components such as buttons, menus, menu items, dialog box options, fields and keywords, as well as menu commands. Bold is also used with '+' to represent key combinations. For example, **Ctrl+Click**

The '>' character is used to denote a menu and sub-menu sequence. For example, **File** > **Open**.

`Monospaced` font is used represent system elements such as command and parameter names, program names, path names, URLs, directory names and code examples.

`Bold monospaced` font is used for commands that must be entered at the Command Line Interface (CLI), For example, **`mctr_gui`**

1.4. Installation

For details on installing the TITAN Executor for the Eclipse IDE plug-in, see the Installation Guide for TITAN Designer and TITAN Executor for the Eclipse IDE [2].

1.5. Reporting Errors

The following information should be included into trouble reports:

- Short description of the problem.
- What seems to have caused it, or how it can be reproduced.
- If the problem is graphical in some way (displaying something wrong), screenshots should also

be included.

- If the problem generates some output to:
- TITAN Console
- TITAN Debug Console
- If the Error view contains some related information, that should be copied too.

Before reporting a trouble, try to identify if the trouble really belongs to the TITAN Executor for the Eclipse IDE plug-in. It might be caused by other third party plug-ins, or by Eclipse itself.

Reporting the contents of the Consoles and the Error log is important as TITAN consoles display the commands executed and their results and the Error log may contain stack traces for some errors. To identify relevant log entries the easiest way is to search for classes whose name starts with "org.eclipse.titan".
The location on which the Error Log view can be opened can change with Eclipse versions, but it is usually found at **Window** > **Show View** > **Other...** > **PDE Runtime** > **Error Log** or **Window** > **Show View** > **Other...** > **General** > **Error Log**.

Chapter 2. Getting Started

2.1. The TITAN Executing Perspective

WARNING

The execution of TITAN Java projects (the Java side of the Test Executor) is done as Eclipse native Java applications. It is not yet fully integrated to the usual interface elements like Views that support the execution of the binaries of the C side of the TITAN Test Executor. For information on executing TITAN Java projects see The Launching Modes Supported by the TITAN Executor Plug-in for TITAN Java Projects.

NOTE

In newer Eclipse versions "Launch configuration" is called "Run configuration", however throughout in this document "Launch configuration" is used.

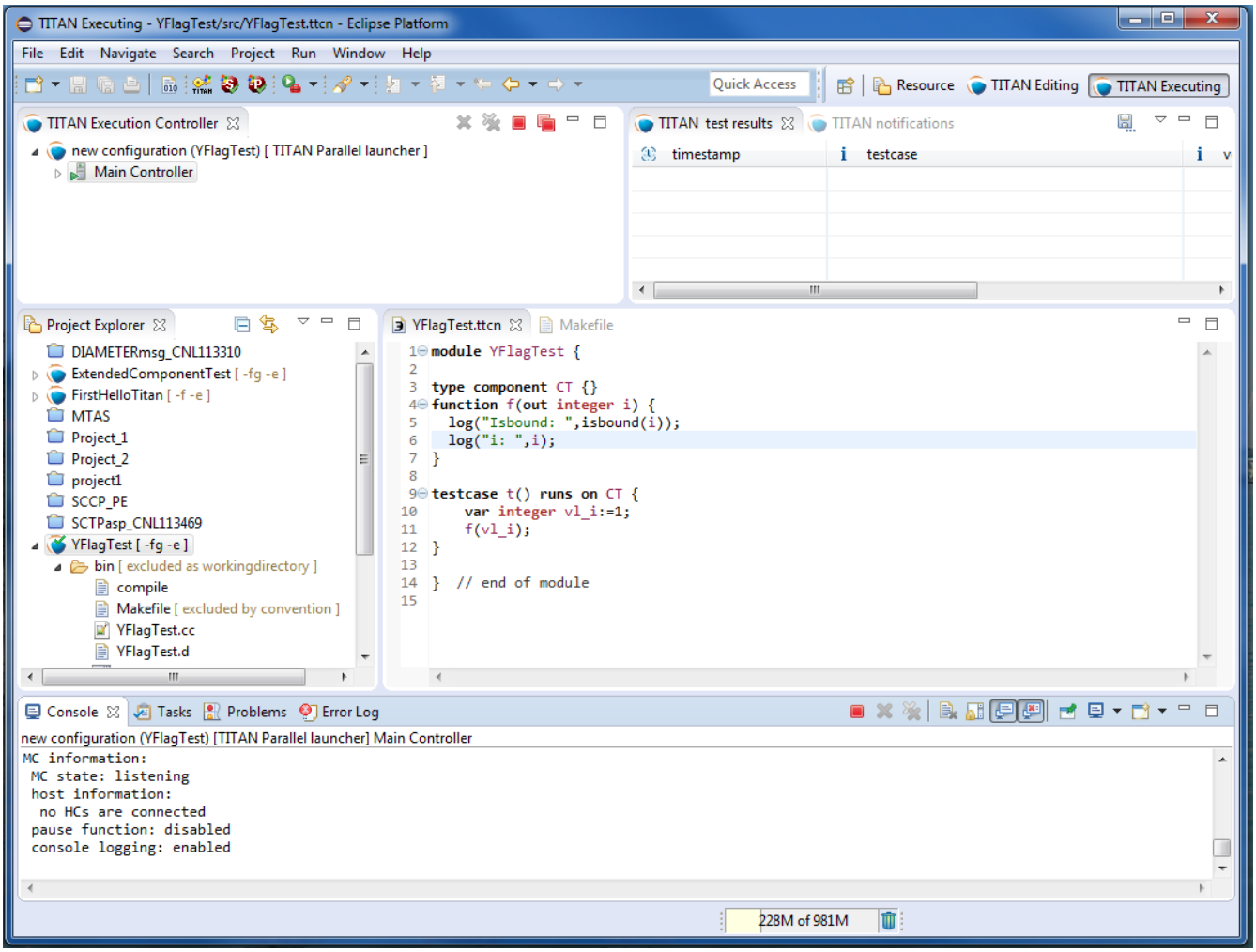


Figure 1. TITAN Executing perspective

The TITAN Executor plug-in provides its own perspective to Eclipse. This is a layout of visual elements that provides a good environment for working with TITAN. This layout is a starting point, since users can create their own layout in Eclipse, to set the best working environment for themselves.

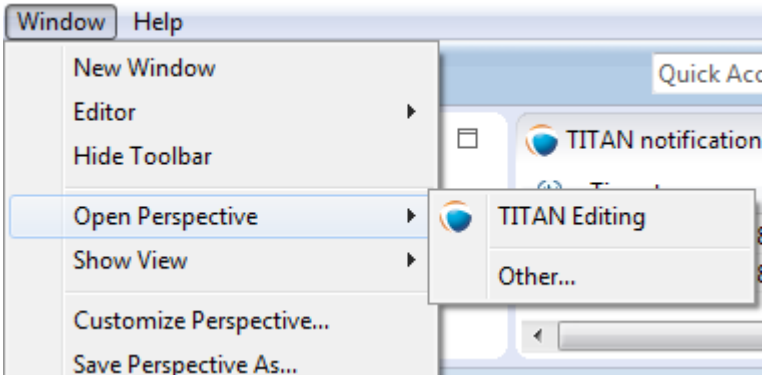


Figure 2. Opening a perspective

Open the TITAN Executing perspective by opening **Window** > **Open Perspective** > **Other...**.

In the pop-up window select **TITAN Executing**.

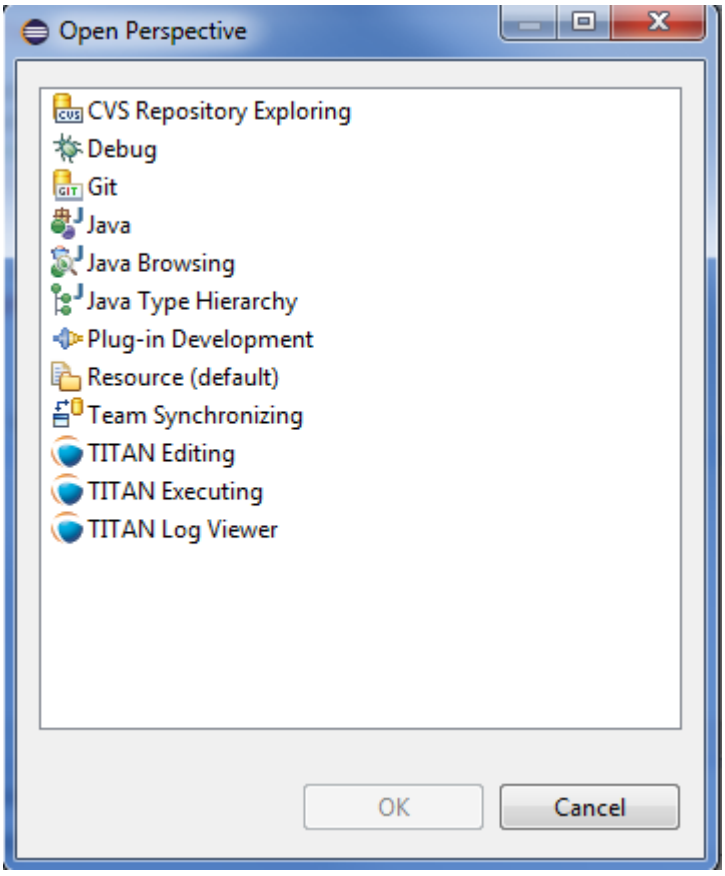


Figure 3. Selecting the TITAN Executing perspective

The layout is shown in Figure 1, and contains the following elements:

- The upper half of the window shows the views defined by the Executing perspective:
- **TITAN Execution Controller**:

  This view allows the users to monitor and control the started executions.

- **TITAN Test results**:

  This view optionally shows the test case verdicts fetched from the notifications and related to the selected executor in the TITAN Executor monitor view.

- **TITAN notifications**:

  This view shows all of the notification messages, error messages and console messages, received from the selected executor in the TITAN Executor monitor view.

- Creating launch configurations and executing them is available in the **Project Explorer view**.
- In the **Editor area** the source codes and configurations can be edited.
- The lower half of the window shows four views:
- **Console**:

  The console shows all related information about build procedure, and execution. Each executor opens a new console page.

- **Tasks, Problems and Error Log**:

  This views are not part of the TITAN Executor, however, shows any Executor related information, for example, errors during a project build.

By default, the **Launch Commands** are enabled in this perspective, as shown in Figure 4. Only the **Run Configuration** is supported by the Executor plug-in. With this tool, new launch configurations can be created, or existing ones modified.



Figure 4. Launch Commands

Chapter 3. Setting Workbench Preferences and Project Properties

3.1. TITAN Executor Preferences

Workbench preferences set user specific general rules, which apply to every project, for example, preferred font styles, access to version handling systems.

Open **Window** > **Preferences**, and select TITAN Executor from the tree.

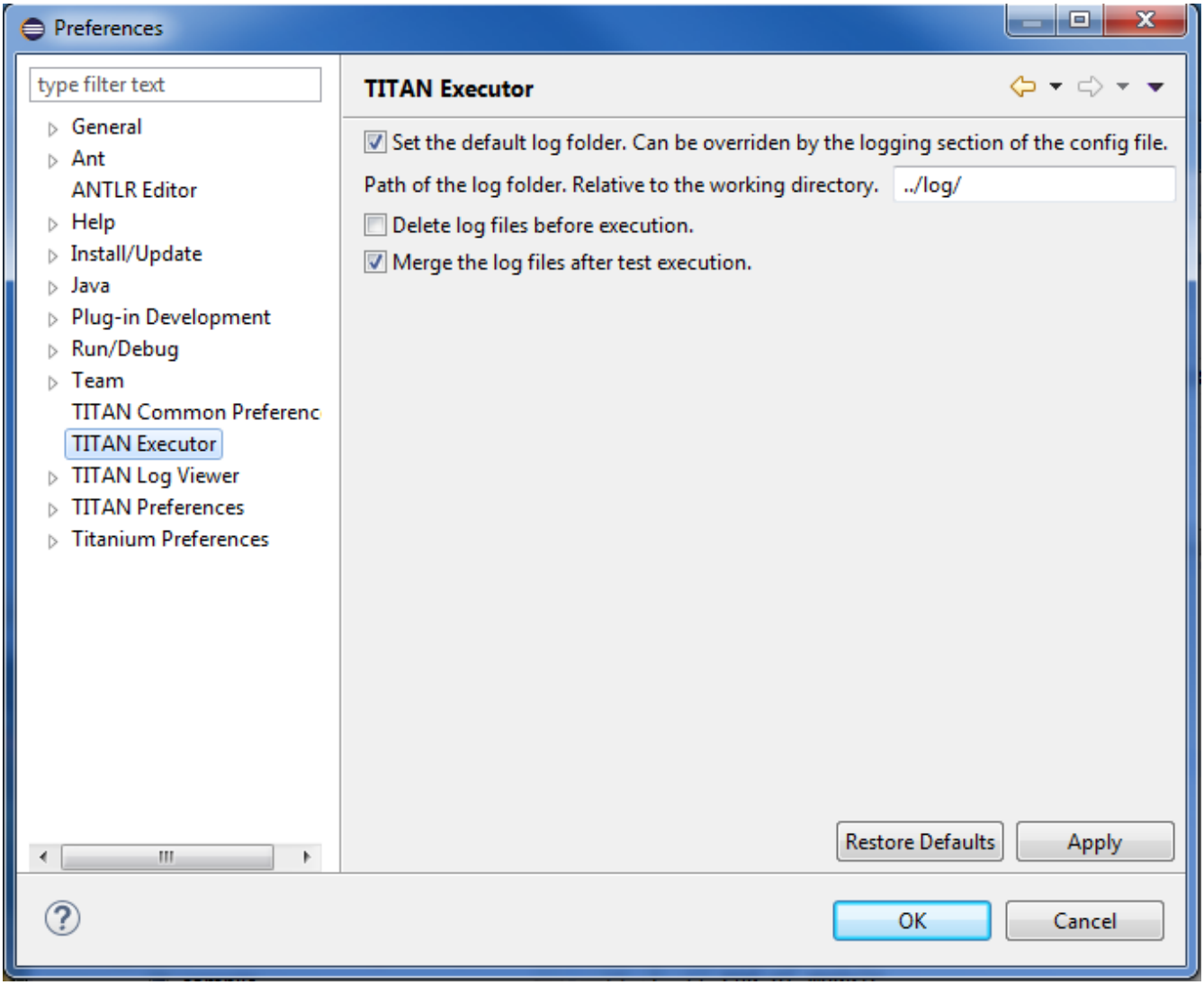


Figure 5. TITAN Executing perspective

The following options can be set on the TITAN Executor preferences page:

- **Set the default log folder.**

  The tests executed by the TITAN Executor will create the log files in the given folder. This option is enabled by default. This option is overridden if the `"FileName"` option is set in the `[LOGGING]` section of the runtime configuration file.

- **Delete log files before execution.**

The log files in the default log folder will be deleted before each test execution. Files with `.log` extensions are considered to be log files. This option is only available if the default log folder has been set, by default it is disabled. If the files cannot be deleted, an error message will be displayed.

- **Merge the log files after test execution.**

  When the test execution is finished, the log files found in the default log directory will be merged into a single file. This option is available if the default log folder has been set, by default it is enabled.

3.2. TITAN Executor Project Properties

To open project properties: **right click** the project and select **Properties**.
On the project property page it is possible to override the workspace settings for the selected project.

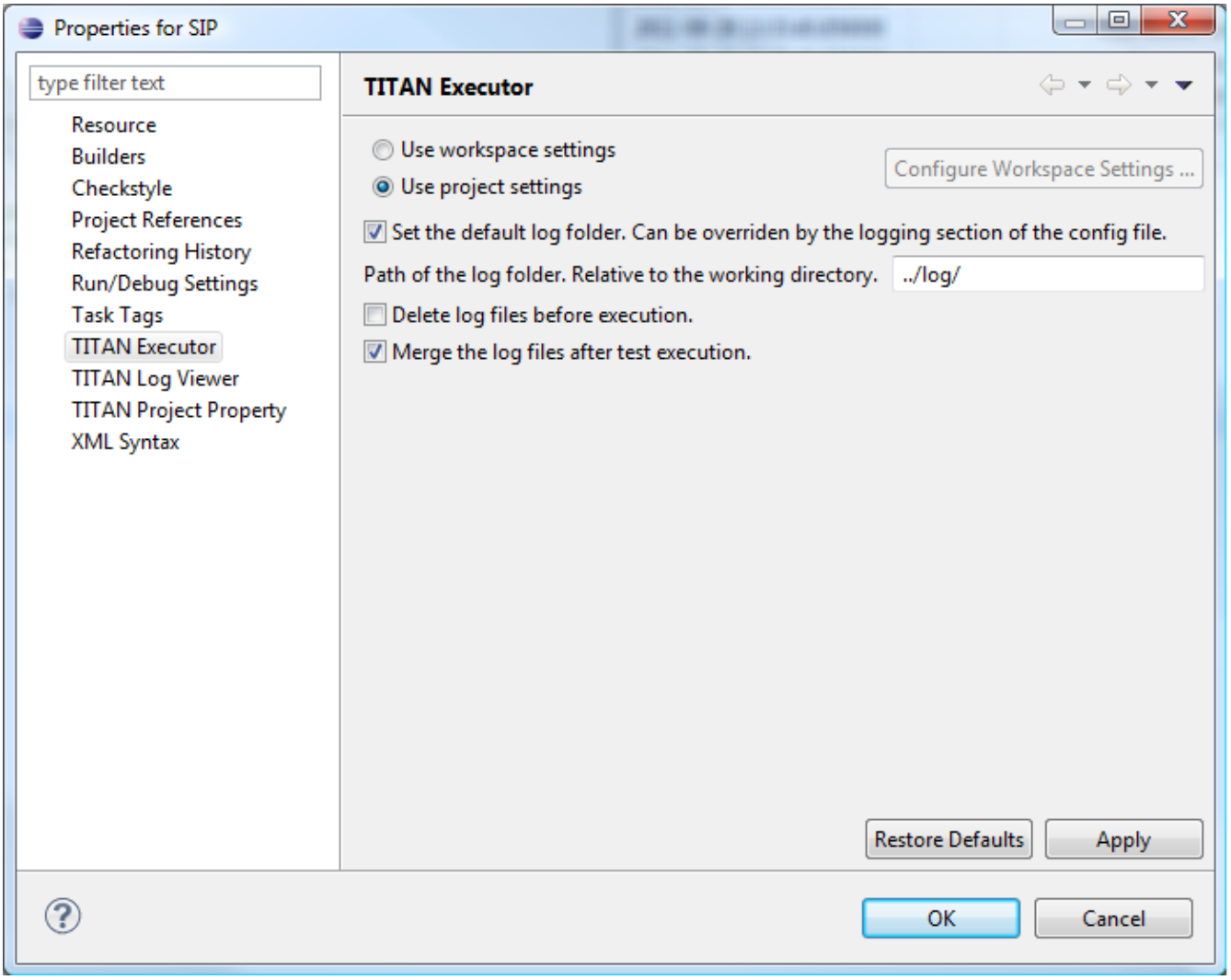


Figure 6. Executor property page

Chapter 4. Launching the Test Suite

This chapter describes launch configurations, their options, and launching modes. After building an executable test suite, it is ready to be launched. In Eclipse, every aspect of the launch can be configured, for example, different environmental settings can be created by creating different launch configurations, without modifying the system environment variables, so different test environments can be created.

NOTE

This chapter discusses only the launching modes related to TITAN C++ projects, i.e. the TITAN Single, the TITAN Parallel and the TITAN JNI launching modes. The TITAN native Java launching mode that is specific to TITAN Java projects is detailed in a separate chapter, see The Launching Modes Supported by the TITAN Executor Plug-in for TITAN Java Projects.

Please note, that TITAN JNI launching mode is marked as obsolete and will be removed in later releases.

4.1. The Launching Modes Supported by the TITAN Executor Plug-in

The TITAN Executor can operate in single or parallel mode.

- The single mode - that is also called non-parallel mode - is thought for TTCN–3 test suites built around a single test component. It is forbidden to create parallel test components in single mode, thus the test suite is not supposed to contain any `create` operation; otherwise the test execution will fail.
- The parallel mode offers full-featured test execution including distributed and parallel execution. The goal of introducing the single operating mode was to eliminate redundancies concerning parallelism, and thereby increase the speed of execution.

It is possible to execute non-parallel test suites in parallel mode, but doing so results in unnecessary overhead. The C++ code generated by the compiler is suitable for both execution modes, there are no command line switches to select mode. The only difference is that another Base Library has to be linked in single and another in parallel mode.

The TITAN Executor plug-in is built on the TITAN Executor and provides support for the following launch and execution modes. These are only available for TITAN C++ projects:

- Single mode:

  This mode executes the built executable, and parses its output for information that can be displayed. There is no limitation on the amount of simultaneously running executions of this kind.

- Parallel mode:

  This mode executes the `mctr_cli` program, and continuously parses its output for information

that can be displayed. Although some functions can be reached from the graphical user interface, the main advantage of this launch mode is that it can be driven from its console window, just as `mctr_cli` could be driven from the command line. The user interface reacts to console outputs by the `mctr_cli`. However, as the user is able to change every aspect of the test execution system from outside the execution monitor view, an always consistent controlling environment cannot be provided. There is no limitation on the amount of simultaneously running executions of this kind.

- JNI mode (*obsolete*): This mode executes the main controller through the JNI interface, and continuously parses its output for information that can be displayed. The functions can be reached from the graphical user interface. The user interface reacts to the state changes of the main controller through a pipe. There is no limitation on the amount of simultaneously running executions of this kind.

NOTE	Execution in JNI mode is not supported on Windows.

- Only one parallel execution can be run at a time.
- The plug-in does not provide any means for the user to execute shell commands in the shell where the execution is happening.
- As the plug-in is directly interfacing with the C/C++ code in the main controller, it is dependent on the version and the platform of main controller.

Note that some execution modes require a properly set up runtime configuration file.

4.2. Creating Launch Configuration

Launch configurations can be created, modified and deleted in the **Create, manage, and run configuration** dialog window. It can be opened in the numerous ways.

4.2.1. Run as...

Right click on a project in the **Project explorer** and select **Run configurations**... from the **Run as** option.

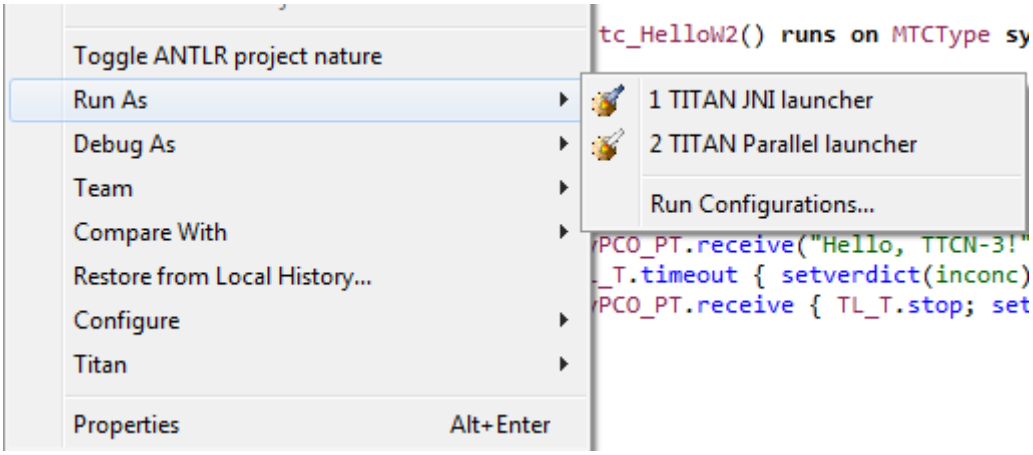


Figure 7. Executing from the Project Explorer

4.2.2. Running from the Launch Command Toolbar

In the toolbar, click the down arrow at the **Launch Commands**, and select **Run as.**
If not visible, modify the Launch properties in the **Windows** > **Customize Perspective**.

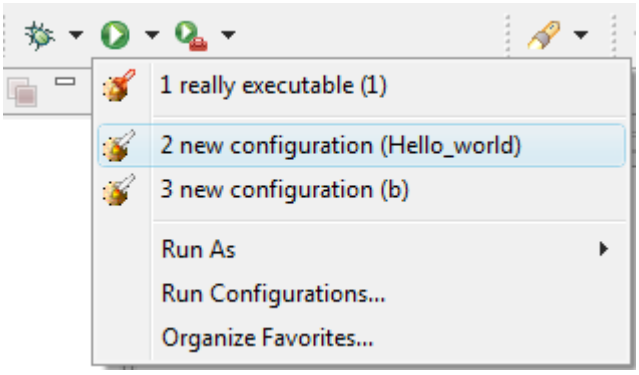


Figure 8. Executing from the Launch Commands toolbar set

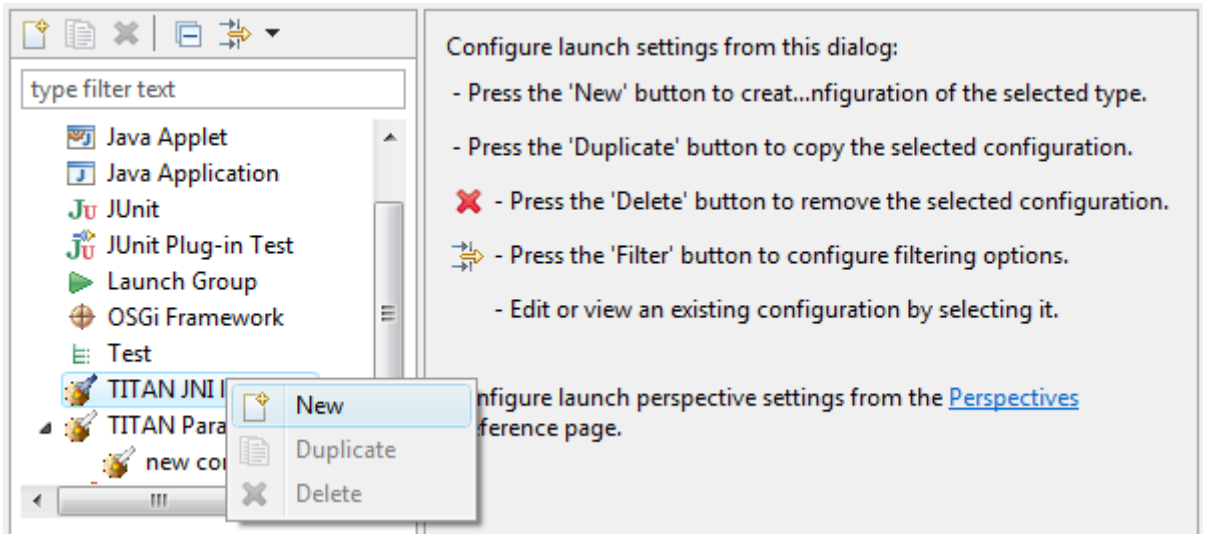


Figure 9. Launch configuration options

The following operations are available:

- '**New**':

  This button can be used to create a new launch configuration. The values of the created launch configurations will be set to their defaults.

- '**Duplicate**':

  This button will create a new launch configuration, filled with the values of the source launch configuration. This button should be used if the new launch configuration does not differ too much from an earlier one.

- '**Delete**':

  This button deletes the actually selected launch configuration.

All launch configuration types supported by the Executor plug-in can be found in the panel.

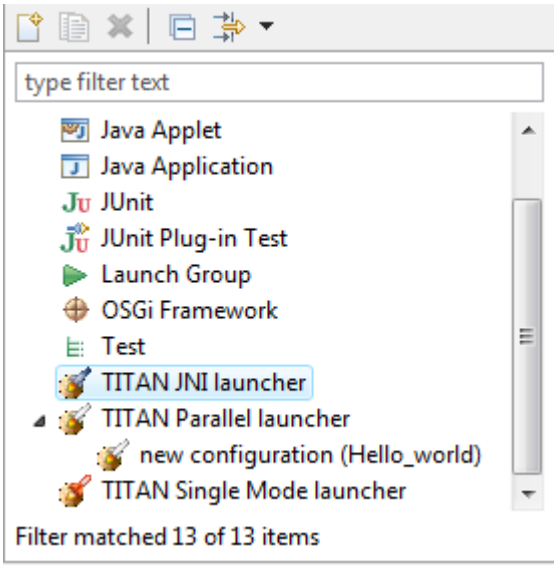


Figure 10. Launch configuration types

Other launch configuration types supported by other plug-ins are also available here.

4.2.3. Creating a Launch Configuration Using Launch Shortcuts

It is possible to create the very first launch configuration for a project, or configuration file in a much easier way then described before:

- Right click on a project in the **Project explorer** > **Run As**
- Right click on a .cfg file in the **Project explorer** > **Run As**

Both will bring up the launch shortcuts available for the selected project.

In the first case the launch configuration will not have any configuration file set by default, if there isn't any configuration file. It will set the only configuration file if there is exactly one configuration file. It will open a list of configuration files belonging to the project if there are more such files.

In the second case the selected .cfg file will be set as the configuration file to use.

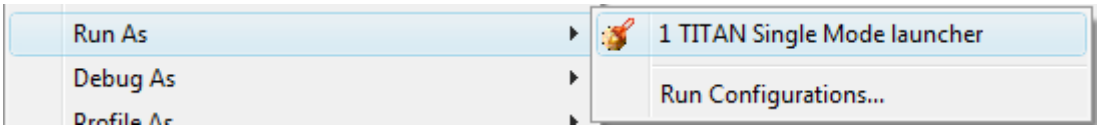


Figure 11. The single Mode execution launch shortcut being active on a project

When this function is invoked on a resource to which no launch configuration has ever been created:

1. A new launch configuration is made.
2. The project, working directory, executable and configuration file paths are initialized.
3. If the executable is found, the available testcase will be extracted.
4. In case of JNI and Parallel mode execution a single default Host Controller is also initialized.
5. Finally the newly created launch configuration is launched automatically.

When this function is invoked on a resource to which there has been already created exactly one launch configuration, that launch configuration will be launched automatically.

When this function is invoked on a resource to which several launch configurations have been made, a list will be displayed for the user to select the one to launch, or cancel to create a new one.

Please note, that after the creation of these launch configuration it is possible to fine tune them just like any other launch configuration using the Launch Configuration Dialog.

4.2.4. Basic Main Controller Options Page of the Launch Configuration

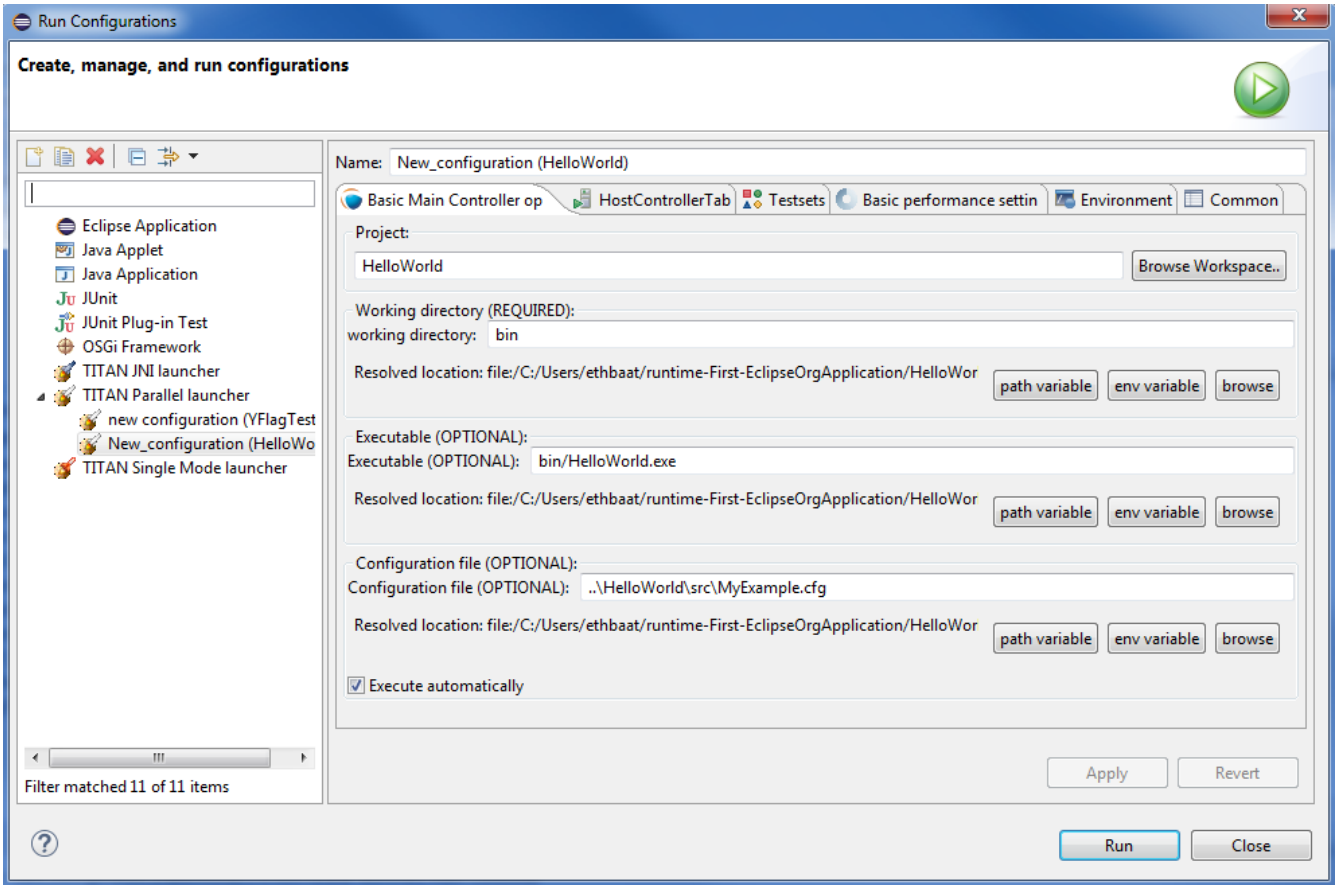


Figure 12. Basic Main Controller options page

On this page it is possible to set:

- The name of the project.

  Filling this field is mandatory. The entered name is checked for validity. The project's root folder is used during the automatically filling of the other fields. The path variables are relative to the project's root supporting the portability of the project as whole. If you enter the name of a valid project with TITAN nature (or select one by browsing, as can be seen below), having the needed build options set, then the fields of the working directory, the executable and the configuration file will be filled in automatically.

NOTE	It is encouraged to use the **Browse Workspace** button to select a valid project from the workspace that simplifies the filling of the other fields, as well as reduces the possible mistakes.

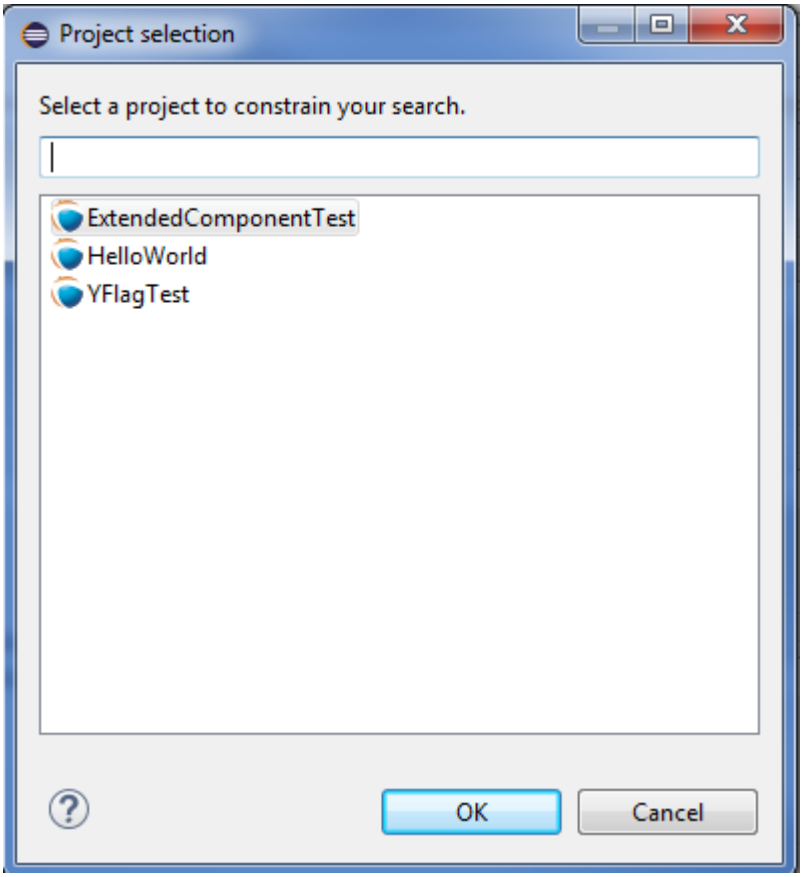


Figure 13. Selecting a project

- The working directory of the project.

  In single mode the built executable and in Mctr_cli mode the Main Controller is executed from this directory. The entered directory path is checked for validity.

- The executable of the project.

  Please note that this executable is used to fill in the list of testcases on the Testsets page, if you change the path, it will be re-checked, and the data for the Testsets page will be re-evaluated. The entered file path is checked for validity.

- The path of the configuration file.

  Please note that not only the existence but also the validity of the configuration file is evaluated here. If a problem was found while trying to process the configuration file, the launch process will be interrupted here. Please note that this evaluation is done every time this configuration page becomes active, meaning that switching to and from this page can take some time. The entered file path is checked for validity.

- Execute automatically

  Whether the user wish to start executing the configuration file automatically when the launcher is started. Please note that this option is turned on by default.

All fields can be filled in either by entering the proper values, or via browsing for them.

4.2.5. Host Controllers Page of the Launch Configuration

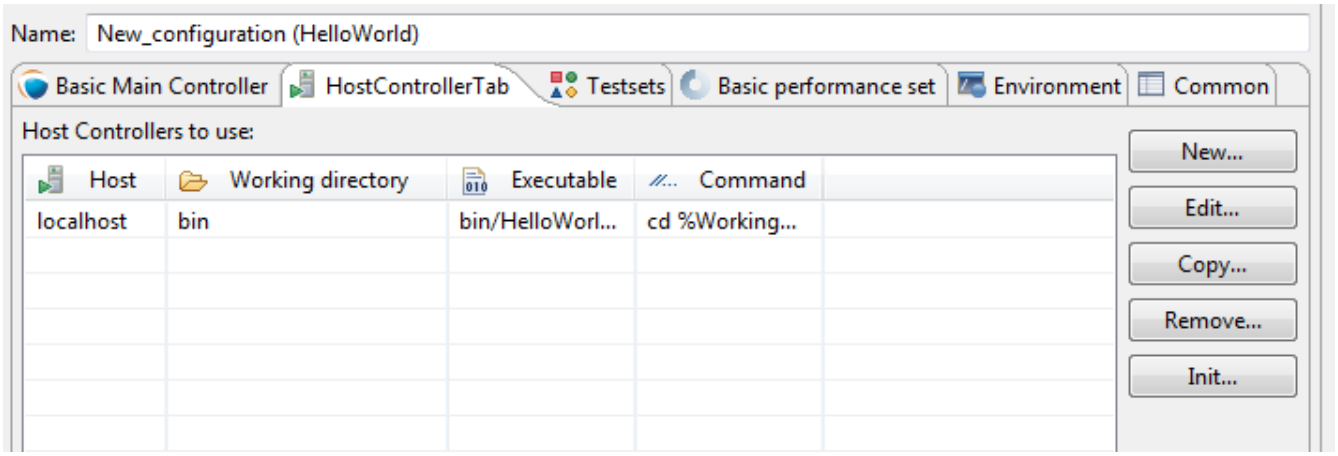


Figure 14. Host Controllers

On this page the Host Controllers can be managed.

When activated in the executors, these Host Controllers will be started and parameterized to connect to the Main Controller automatically. Please note that other host controllers might also connect to Main Controller, but those must be manually parameterized.

There are four operations available on this page:

- **New...**:

  With this button a new Host Controller can be created.

- **Edit...**:

  With this button the settings of an existing Host Controller can be changed.

- **Copy...**:

  With this button a copy of an existing Host Controller can be created.

- **Remove...**:

  With this button an existing Host Controller can be removed.

- **Init...**:

  Pressing this button will remove the existing host controllers and try to automatically create one based on the settings of the project (provided on the Main Controller page, for more information please refer to section 4.2.4)

The first two of these options opens up the Host Controller dialog (below).

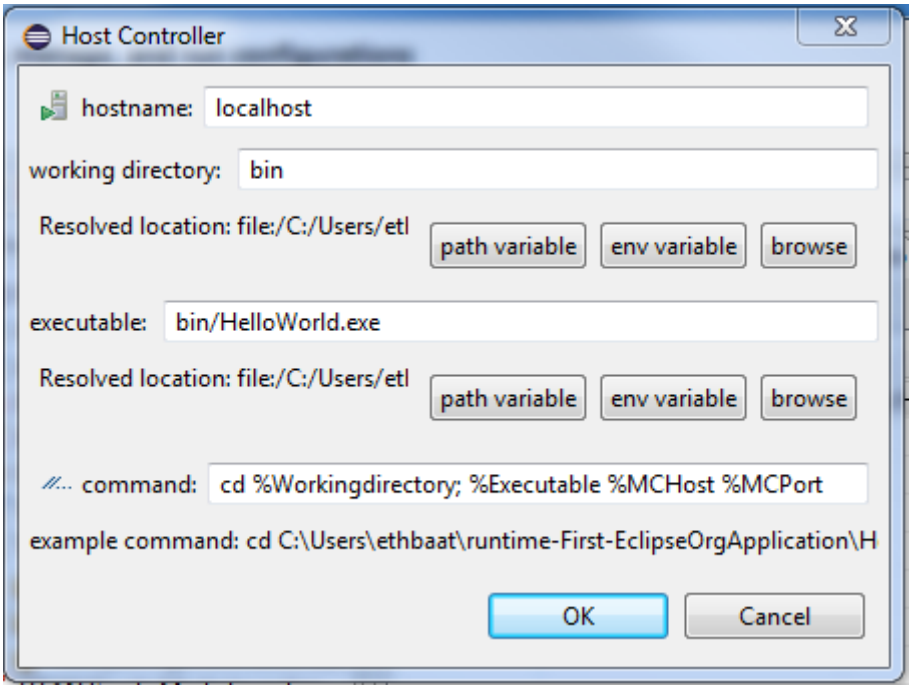


Figure 15. Host Controller dialog

On this Dialog the following options can be set:

- the name of the host
- the working directory of the host
- the executable on the host
- the command to execute when starting a given Host Controller

Please note that:

- none of the fields is required to be unique
- only the command to execute is required for successful operation
- the name, working directory and executable fields are only presented to ease the creation of Host Controllers, especially copying them. In this case it is possible to use the exact same parameterized command for several Host Controllers

You can enter special 'macros' into the command, which will be extracted just before executing the command.

- `%Host` is replaced by the contents of the name field of the host
- `%Workingdirectory` is replaced by the contents of the Working directory field
- `%Executable` is replaced by the contents of the Executable field
- `%MCHost` is replaced by the address where the Main Controller is running
- `%MCPort` is replaced by the port on which the Main Controller is accepting connections from the Host Controllers

Please note that the values for the last two macros are provided by the Main Controller.

The meaning of the default command string:

- `rsh %Host` : means that before starting the Host Controller it is required to login to a remote machine.
- `cd %Workingdirectory;` : means that the Host Controller will be started from a specific working directory, and all log files will generated in this directory.
- `./%Executable %MCHost %MCPort` : means that the Host Controller is to be executed with the 2 parameters describing how to connect to the Main Controller.

NOTE	The `./`, executing on this way is only required if the location of the Host Controller is provided without a full path. In case a full path is used, this part must be removed.

Please also note that in Single launch mode this page does not exist, as in single mode Host Controllers cannot be used.

4.2.6. Testsets page of the launch configuration

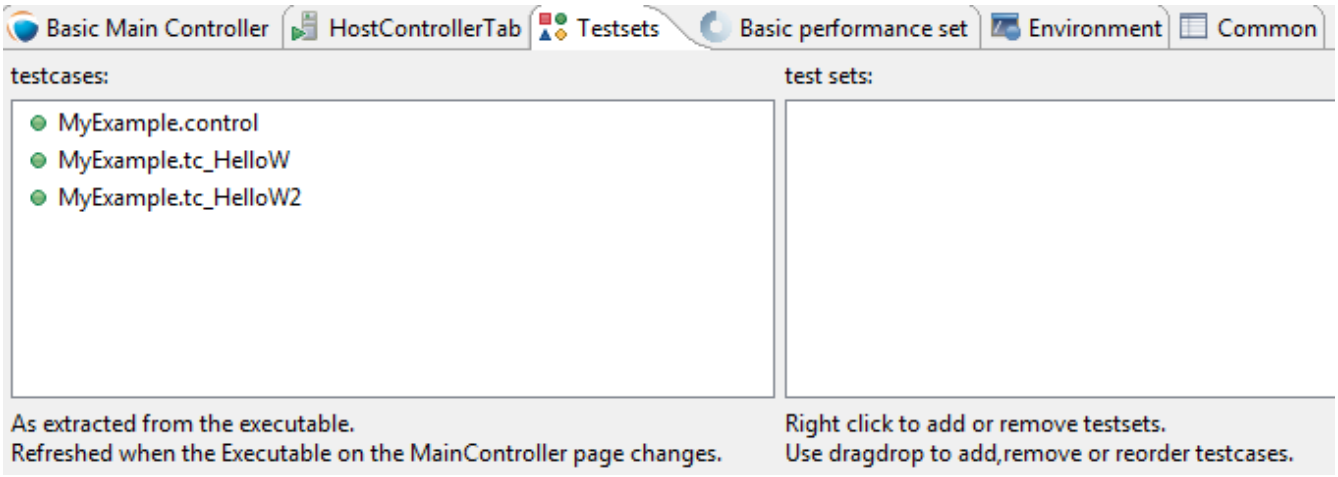


Figure 16. Testsets page

On this page testsets can be set and managed.

The page consists of two areas:

- On the left side is the testcases panel.

  The available testcases and control parts are listed here. They are collected from the executable provided on the Basic Main Controller page.

- On the right side is the testsets panel.

  The already created testsets and their contents are displayed here.

To reach the basic test set operation **right click** on the testset panel.

Operations on the testsets panel:

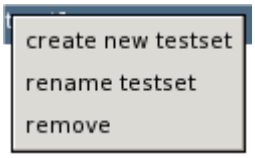


Figure 17. Basic testset operations

- **Create new testset**:

  By clicking on this action a new testset can be created. Please note that the names of testsets must be unique.

- **Rename testset**:

  By clicking on this action while a testset is selected, it can be renamed. Please note that the names of testsets must be unique.

- **Remove** (testset):

  By clicking on this action while a testset is selected, it can be removed.

- **Remove** (testcase):

  By clicking on this action while a testcase is selected from a testset, it can be removed from the testset.

Addition, reorganization and copying of testcases is supported via drag & drop operations.

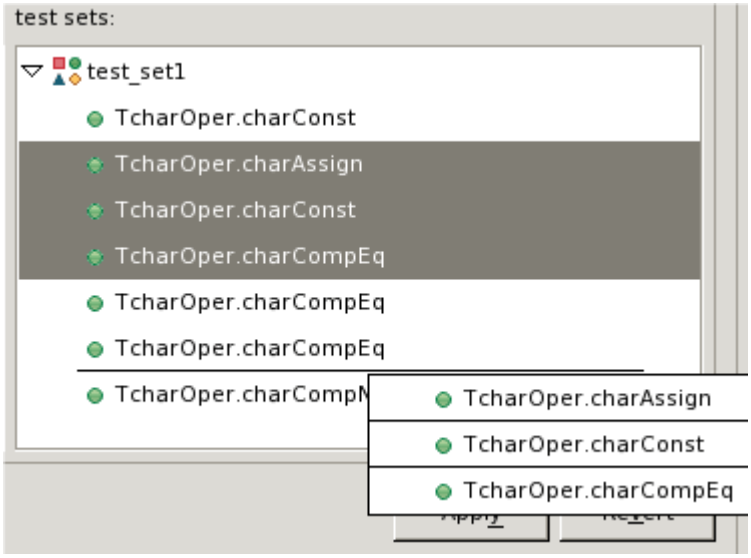


Figure 18. Drag & Drop on the testsets page

- Inserting new testcases in a testset:

  To insert a set of testcases into a testset, they should be grabbed from the testcases panel and dropped on the desired testset.Please note that they can be dropped right to their intended positions if the testset is in extracted state. If the testset is in closed state, it can be opened without disrupting the drag & drop operation by holding the mouse over the testset.

- Moving testcases:

To move a set of testcases into another position select them, and use drag & drop to move them to the desired place.

- Copying testcases:

  Copying testcases is almost the same as moving testcase, with the only difference being that the copy type of drag & drop must be used.Please note that on most platforms this behavior can be activated by holding down the **Control button on the keyboard** while the drag & drop operation is ongoing.

NOTE	It might happen that the executable was changed since the testsets were last modified, in a way that some testcases contained in testsets were removed. In this case the structure of the testsets is not adapted automatically; rather it displays warning signs before the missing testcases and the testsets containing such testcases, as it can be seen below.

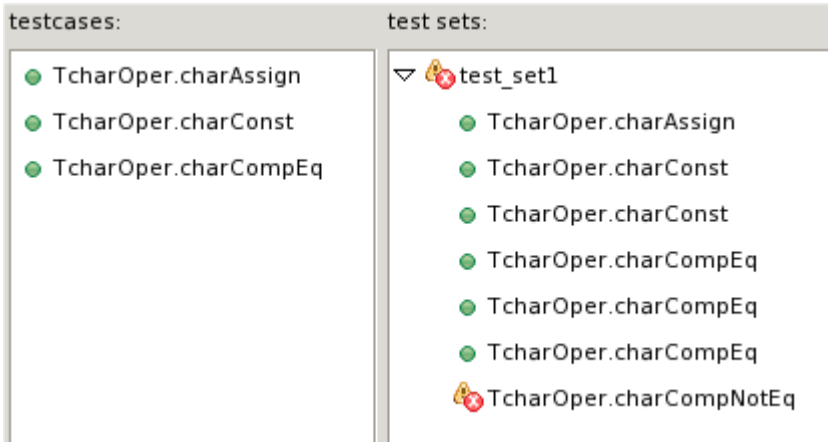


Figure 19. Erroneous test set

Please note that all three launch configuration types supported by the Executor plug-in use this page the exact same way (however they might use the created test sets differently).

4.2.7. Basic Performance Settings Page of the Launch Configuration

On this page performance affecting options can be set. Please note that there are only two general settings available, all other settings are launch mode specific, as it can be seen on figures Figure 20, Figure 21 and Figure 23.

- General performance options:

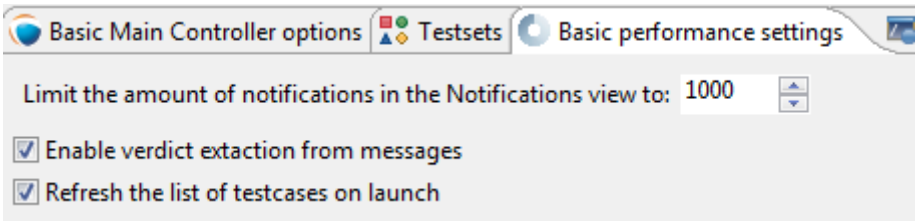


Figure 20. General part of the performance page

- Limiting the amount of notifications:

With this option the maximum amount of notification messages, which can be kept available in the notification view can be set. Basically it can be used in two ways:

- Setting it to 0:

  This means, that every notification message (console logs and error messages) will be kept available. Please note that in a lengthy execution, the amount of used memory might become very high. Please also note that the views refreshing speed might depend on the amount of elements, which need to be redrawn.

- Setting it to a positive number:

  This means that the maximum amount of notification messages that are accessible will be around this amount. Please note that if older messages are not needed this is a good way to decrease memory requirements, and possibly increase execution speed. Please also note that the real amount of accessible notifications might somewhat exceed this threshold. This is because of performance reasons, as removing several elements at once is much faster than removing elements one by one.

- Verdict extraction from notification messages:

  If this option is set, then the notification messages are parsed for possible verdict setting messages. This allows using the TITAN test results view, where only such verdict setting information is displayed. However, please note, that this requires a regular expression matching for every message, which can slow down the execution.

- Refresh the list testcases on launch:

  If this option is set right after the launch of a launch configuration (but still before actually executing something) if the binary is set, it will be contacted for the actual list of testcases and control parts. The elements from this list which are not included in the configuration (because they were added later) will also be displayed in the execution dialog.Please note that this will not update the configuration itself, as that could lead to data loss regarding testsets (containing the testcase temporarily removed).

- Specific performance options for Single launch mode (below):

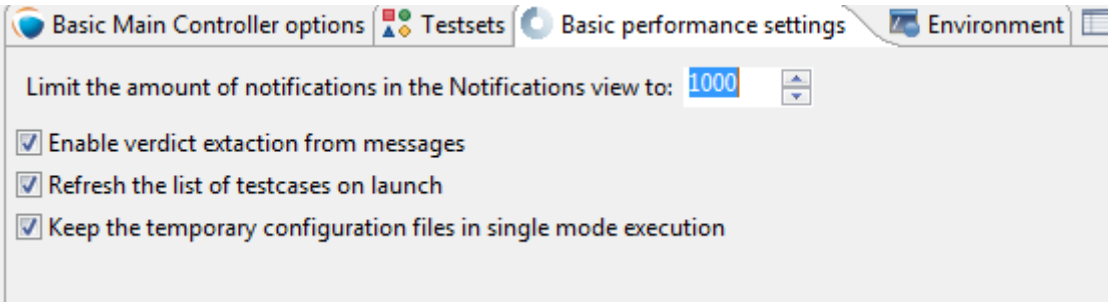


Figure 21. Performance page of the Single launch mode

- Keeping temporary configuration files:

  In single launch mode the execution is controlled with temporarily created configuration files. For example, if a testset is to be executed, the executable will be called with a configuration file

containing a properly generated Execute section.

- If this option is not set, then temporary configuration files are deleted after each execution, to save disk space.
- If this option is set, then the temporary configuration files are kept after execution. This can be used to create specific configuration files automatically.
- Specific performance options for parallel launch mode(below):

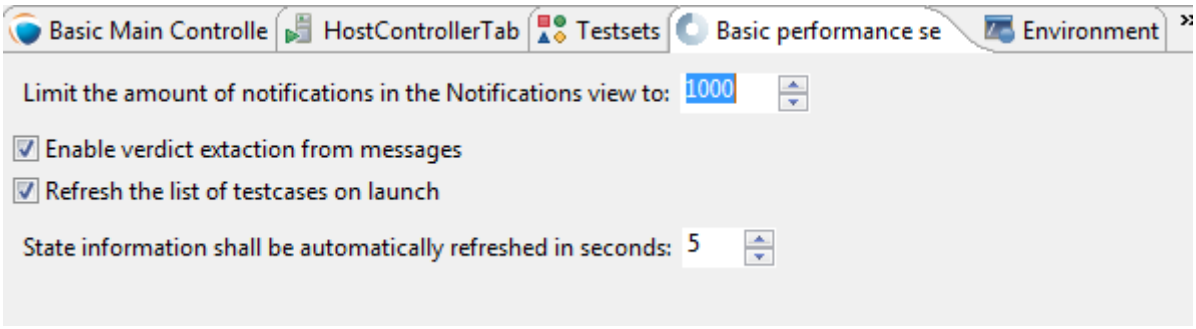


Figure 22. Performance page of the mctr_cli launch mode

- State information refresh:

  In parallel mode the execution is controlled with command line messages and outputs. This means that the eclipse side of the executor and the command line side of the executor has a high probability of being in different states. This option set how often should synchronization be done.
  Please note that values are only accepted in the 1..10 range. Lower values could provide too much load, higher values would could effect execution times.

- Specific performance options for JNI launch mode (below):

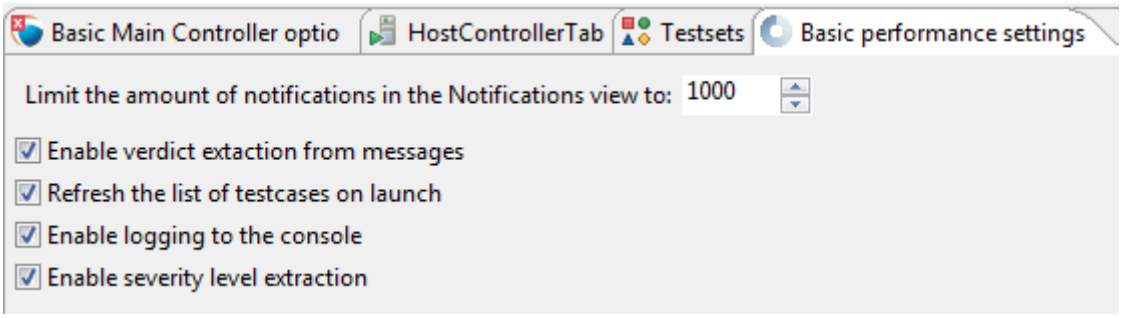


Figure 23. Performance page of the JNI launch mode

- Enable logging to the console:

  If this option is not set, the notification messages will not be printed to the console (but they will still be displayed in the notification view).

- Enable Severity level extraction:

  The extraction of the severity level of events does not really cost too much performance, so it should always be set unless performance is crucial, and this information is not really needed.

4.2.8. The Environment Page of the Launch Configuration

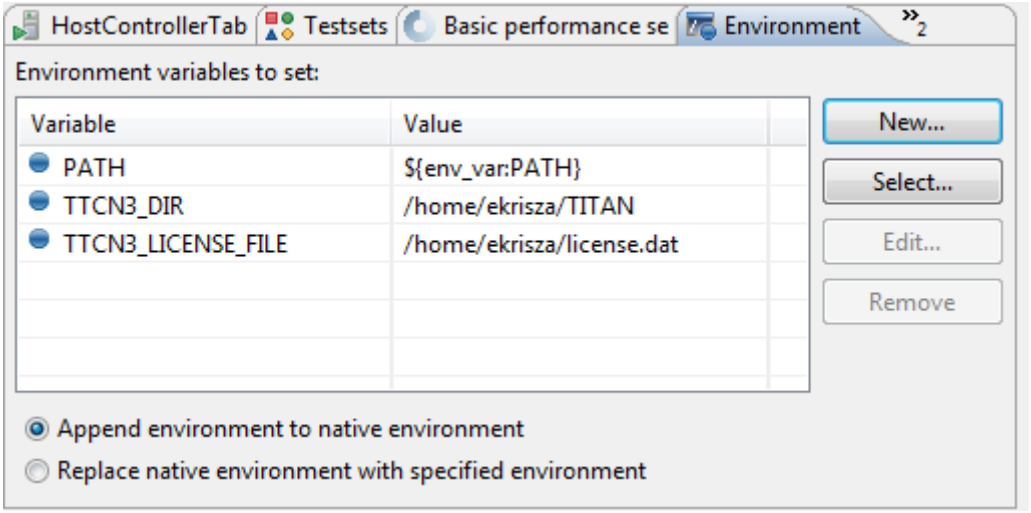


Figure 24. Environment settings page

In general, the environment settings page is the place where the execution environment should be configured.

On this page you can:

- Add new variables with the "**New...**" button.
- Add environmental variables from the actual environment with the "**Select...**" button.
- Modify already set variables with the "**Edit...**" button.
- Delete already set variables with the "**Remove...**" button.

On the bottom of the page there is one more important option: whether you want to append or overwrite the list of variables coming from the operating system, with these variables.

Important:

- Please don't forget that the TITAN executor requires the setting of some environmental variables, to work properly. For more information please refer to the Programmer's Technical Reference [4].
- Please note that the execution is not taking place in the operating system's environment, but in the Java Virtual Machine's environment. Special caution is required, if your tests depend on the values of environmental variables defined in the operating system.
- In order to make dynamic linking work, the `LD_LIBRARY_PATH` environment variable is set automatically for all launch modes. It generally means, that `${TTCN3_DIR}/lib` and the working directories of all reachable projects are appended to `LD_LIBRARY_PATH`. (The user defined `LD_LIBRARY_PATH` always comes first, if it's available.)

Please note that this page is fully provided by Eclipse, however features like appending the environmental variables or using variable variables (this feature can be reached by pressing the **Edit...** button), only take affect if they are implemented in the plug-in, too. Appending or overwriting environmental variables is fully supported. Some effort was made to support the variables too, but as their number, and ways of behaving is internal to Eclipse (meaning that it can

be changed at any time), their usage is NOT RECOMMENDED.

Differences of the launch modes:

- In single mode:

  The shell created to run the built executable will receive the environmental variables.

- In parallel mode:

  Both the Main Controller and the Host Controllers will be executed in shells, having the provided environmental variables set.

- In JNI mode:

  The Host Controllers will be executed in shells, having the provided environmental variables set.

4.2.9. Common Page of the Launch Configuration

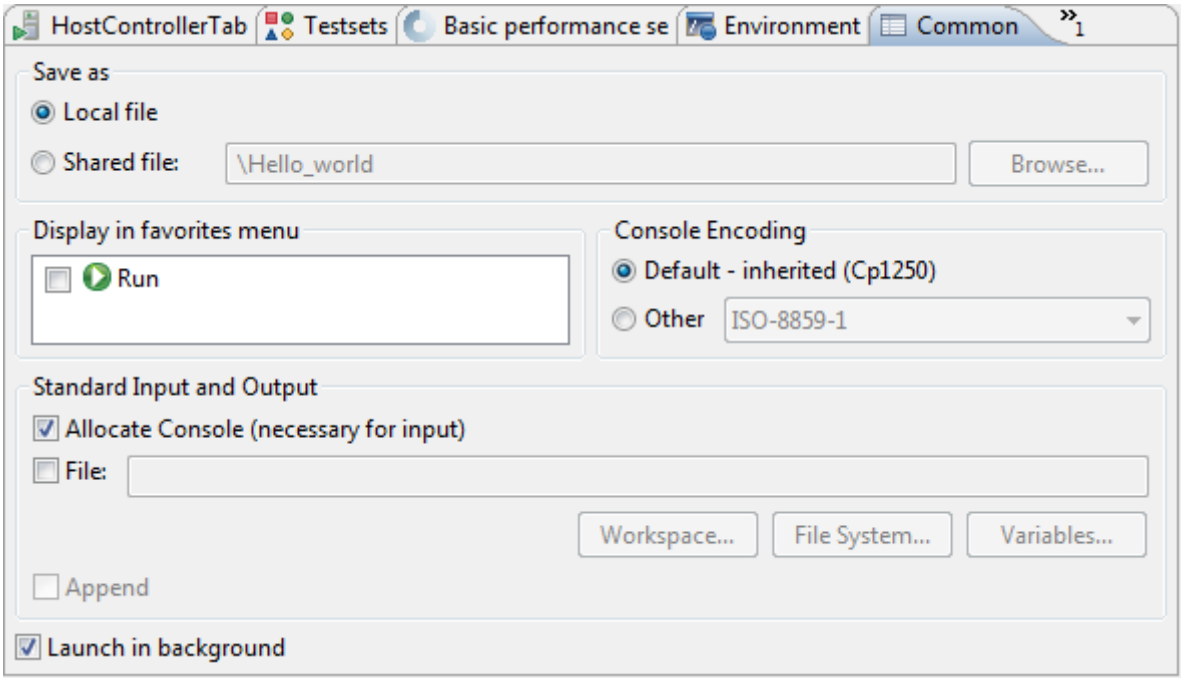


Figure 25. Common page

This is a fully Eclipse provided and supported page.

For us the two most important parts of this page are:

- The Save as region:

  Here you can select a directory where the data of the launch configuration will be saved. The file will be named after the name of the launch configuration, with the extension "launch". Eclipse is automatically taking care of such files, if it finds such a file anywhere in the directories of the projects, it will be offered to the user.

NOTE	Please note that for this reason it is advised to put the launch configurations in the project's directories. Please note that if you choose not to save the launch configuration it will still be saved, but to an internal point in Eclipse's inner data hierarchies. Please note if a project is closed, the launch configurations belonging to it won't be displayed.

- Standard input and output:
  - **Allocate Console**:

    This option should always be checked; otherwise Eclipse will not create a console for the executed processes, disabling communication with them.
  - **File**:

    If a file is set, then every command entered in, or output on the console will be written to the given file, too.
  - **Append**:

    If this option is not set, then every time a new process is started it will erase the contents of the above mentioned file, before writing out the new messages.

NOTE	In JNI mode launch the Standard input and output handling part is not supported.

Chapter 5. Executing and Controlling the Execution of Test Suites

The three different launch modes related to TITAN C++ projects that are supported by the Executor plug-in, also imply three different execution mechanisms (executions and executors).

As the general user interaction interfaces, are about the same, we will discuss them one by one, detailing the differences between the execution modes when they apply.

5.1. Execution Control

Execution can be started on several ways:

- Pressing the **Run** button on a launch configuration, starts an execution configured with that launch configuration.
- Pressing the **Run** button on the launch commands actionset, on the toolbar, will start an execution configured with the launch configuration used the last time.
- Pressing the arrow to the right from the Run button on the launch configuration actionset, and selecting the preferred launch configuration, will start an execution configured with the selected launch configuration.

The launching modes are explained in detail in the chapter describing launching modes supported by the TITAN Executor plug-in.

Please note that redrawing the supported views might cost some performance, for this reason it is supported to close these views. If these views are closed then the performance penalty of redrawing them does not apply, however in that state you cannot control the execution of the tests, and you don't receive any status information. If in the middle of the execution the views are opened up again, they will show the actual state of the system, and you can again take control of the execution. Please note that only the views can be closed; exiting Eclipse stops every ongoing execution automatically.

As soon as an execution is started it will be displayed in the Executor monitor view.

5.2. TITAN Execution Controller View

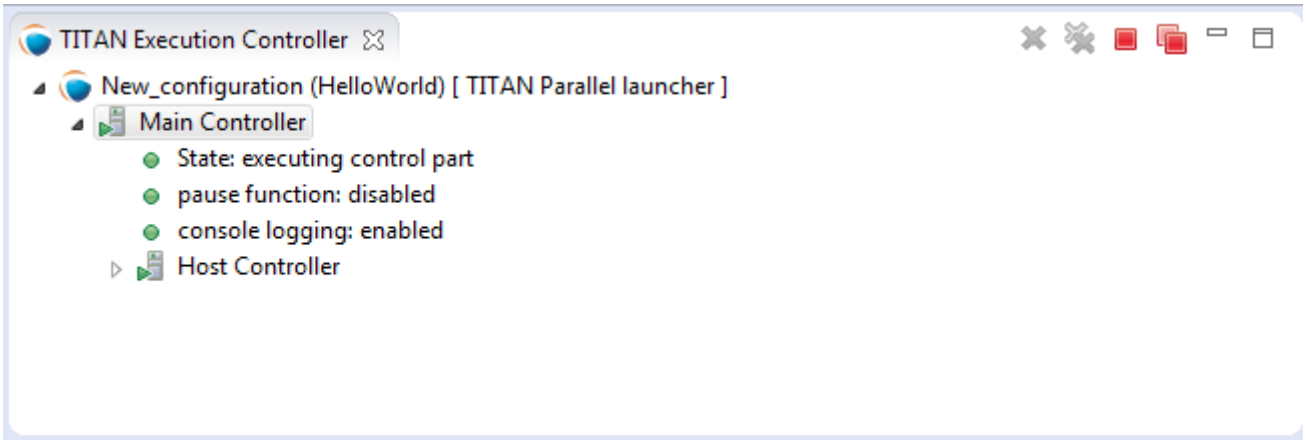


Figure 26. TITAN Execution Controller view

The TITAN Execution Controller view is the place where you can control the execution via their executors.

It is made up of two main parts:

- The toolbar:

  This is the place where general execution mode independent actions are made available.

- The main window:

  This is the place where executor specific information and actions are made available.

Four actions are available from the toolbar:

- **Remove selected terminated launch**:
  This action can remove an already terminated launch from the list.
- **Remove all terminated launches**:
  This action removes every terminated launch from the list.
- **Terminate selected launch**:
  This action can terminate a not yet terminated launch.
- **Terminate all launches**:
  This action can terminate every not yet terminated launch.

These actions are only enabled if they can be applied.

Please note that it is a good idea to look around in the **Window** > **Preferences** > **Run/Debug** section, as several Eclipse level preferences can be set here, that might affect the user experience.

For example:

- In **Console** the preferences of the launch consoles can be set.
- In **Launching** you can choose to have a build before every launch (the build actually only happens if it is required).

The main window contains information about the launches and executors.

This representation has:

- A general part:

  This is a tree, where the root represents the launch, and the leaf represents the executor.

- The name of the root follows the `<name of the launch configuration> [ <launch mode> ]` convention.
  The root always offers all four actions known from the toolbar (they can be reached by a **right click on the root element)**.
- The name of the leaf is always Main Controller.

  The leaf allows the **terminate selected launch** and the **remove terminated launch** actions.

  - A specific part:

    The actions generally provided by the leaf are extended with executor specific ones. In case of the parallel and JNI launch modes, detailed information about the Main Controller can be made visible as a sub tree of this tree node.

Specific commands:

- in Single mode:

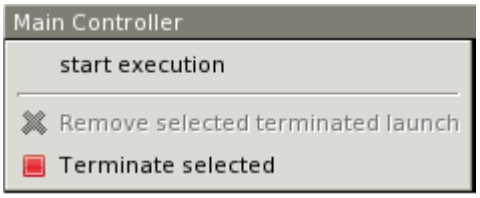


Figure 27. Commands in single mode

- Start execution:

  This action lets you execute your test campaign, via the Execute dialog (Figure 31). Please note that temporary configuration files, based on the selection in the Execute dialog, are generated to drive the execution.

- in parallel mode:

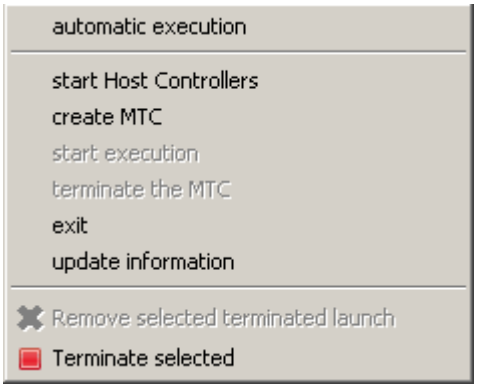


Figure 28. Commands in parallel mode

- **Automatic execution**:

  This action allows you to automatically execute tests. You will be presented with the Execute dialog (Figure 31). After selecting the element you wish to execute, if the launch was configured properly, it will automatically navigate through the steps that are required for the execution.

- **Start Host Controllers**:

  This action starts the Host Controllers set on the Host Controllers page of the launch configuration. Please note that the variables inserted into the Host Controllers commands are extracted at this point. For more information on the Host Controllers page of the launch configuration, please refer to Host Controllers Page of the Launch Configuration;

- **Create MTC**:

  This action executes the "**cmtc**" command, creating the Main Test Component of this execution;

- **start execution**:

  This action brings up the Execute dialog (Figure 31), and executes the "**smtc**" command properly parameterized, with the selected item to execute;

- **Terminate MTC**:

  This action executes the "**emtc**" command, terminating the Main Test Component;

- **exit**:

  This action executes the "**exit**" command, exiting from the Main Controller;

- **info**:

  This action executes the "**info**" command, providing inner state information to the user.

NOTE	The last five commands are commands of the `Mctr_cli`.

All of the commands of **Mctr_cli** can be issued directly from the Console too (including the ones mentioned). The executor will try to adapt to the changes, for example if you select the **info** action, or execute the **info** command in the Console, the information displayed under the executor node will be updated (this can be seen on Figure 29). For more information on the commands of the **Mctr_cli** and how to execute testcases in it please refer to Section 4.4 of the User Guide [3].



Figure 29. Example information display in the Parallel launch mode

- JNI mode:

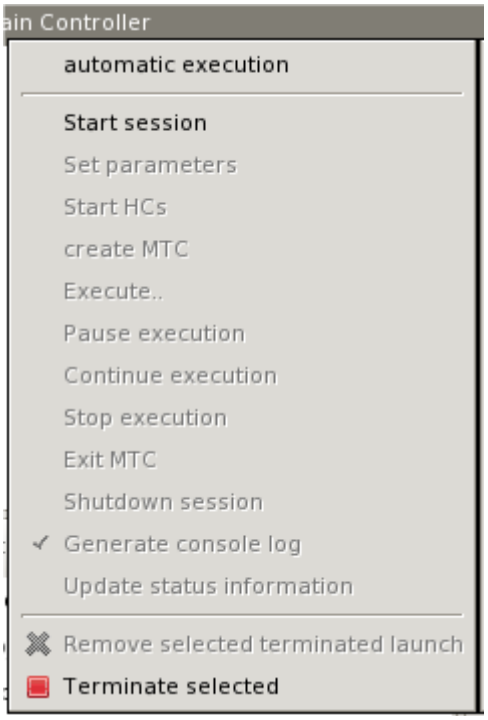


Figure 30. Commands in JNI mode

- **Automatic execution**:

  This action allows you to automatically execute tests. You will be presented with the Execute dialog (Figure 31). After selecting the element you wish to execute, if the launch was configured properly, it will automatically navigate through the steps that are required for the execution.
  If you wish to do these steps yourself, then you can use the following actions.

- **Start session**:

  The Main Controller starts listening for incoming Host Controller connections on the TCP port defined in the configuration file.

- **Set parameters**:

  The Main Controller downloads the configuration file to the connected Host Controllers, so they can process it.

- **Start HCs**:

The Host Controllers, defined on the Host Controller page of the launch configuration, are started. They first try to establish a TCP connection to the Main Controller and then wait for further requests. If a Host Controller connects to the test system after the parameters were already set, the Main Controller will download the configuration file to this new Host Controller, too.

- **Create MTC**:

  Creates the Main Test Component and establishes a control connection between the Main Controller and the Main Test Component.
  Please note that there can be only one MTC in the test system.

- **Execute...**:

  Brings up the Execute dialog (Figure 31), where control parts, testcases, test sets and even execution schemes created in the configuration file can be executed.

- **Pause execution**:

  Sets whether to interrupt test execution after each test case, or not. The actual value is displayed as the checked status of this action (if it is set, then a checked state is displayed). If this action is checked and the actual testcase is finished, the execution is stopped until the **Continue execution** action is selected. If this action is not checked and the actual test case finished execution, then the execution continues with the next test case.

- **Continue execution**:

  Resumes the interrupted test execution.

- **Stop execution**:

  Terminates the test execution.
  The verdict of the actual test case will not be considered in the statistics of the test suite.

- **Exit MTC**:

  Terminates the Main Test Component.

- **Shutdown session**:

  Shuts down the session, terminating the Host Controllers and the Main Controller.
  Please note that the already connected Host Controllers cannot be terminated till this point in the execution.

- **Generate console log**:

  Enables / disables console logging.
  If this action is not checked, console messages will not be generated.
  Please note that in this case the notification messages (originally also considered as console messages), will not be emitted by the Main Controller, this way such messages will be missing from the Notification view too.

- **Update status information**:

  Selecting this action you can update the detailed information, which the main controller provides (an example can be seen on Figure 29).

5.2.1. Execute Dialog

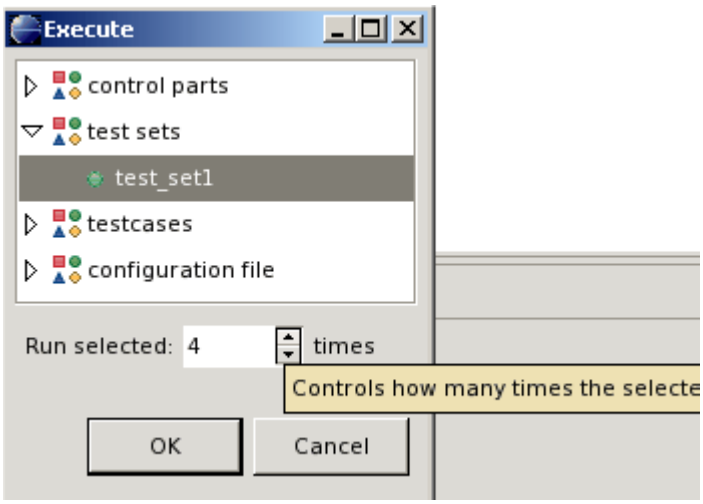


Figure 31. Execute dialog enabled

This dialog (above) is shown by all 3 launch modes when you select to execute tests of any kind.

This dialog represents the executable test elements in a tree:

- Control parts.
- Test sets.
- Testcases.
- Configuration file:

  This means the tests and their order defined in the execute section of the configuration file, provided on the Basic Main Controller options page of the launch configuration (for more information please refer here).

On this dialog you can also select how many times you wish to execute the selected element.

Selecting the amount of execution times is only available if an executable element is selected. If one of the main elements (branches) of the tree is selected, then the execution amount adjusting part of this dialog becomes disabled (this can be seen below).

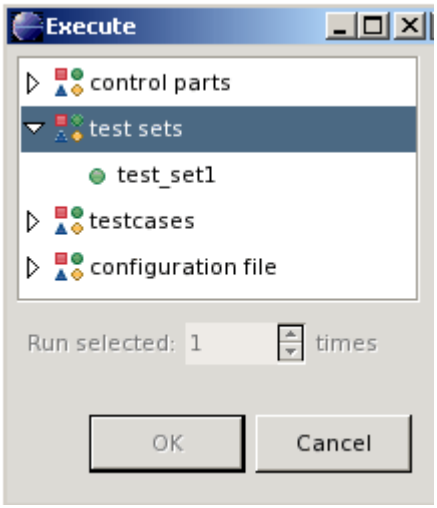


Figure 32. Execute dialog with disabled execution times part

Please note that if an element type is not present then the corresponding branch cannot be expanded. For example if no configuration file was set on the Basic Main Controller options page, then the configuration file branch does not have leafs.

5.3. TITAN Notifications View

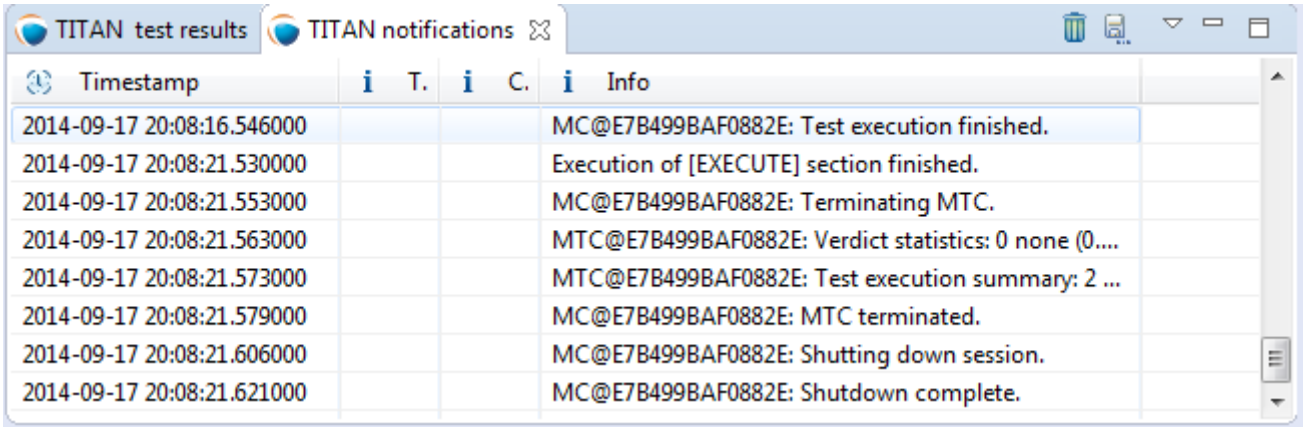


Figure 33. TITAN Notifications view

The TITAN Notifications view (above) contains all of the notifications (previously console and error messages), that might come from the test system.

As several executions can be ongoing at any given time, this view always shows the notification messages, created by the execution/launch actually selected in the TITAN Execution Controller view (Figure 26). For this reason the tool tip of this view shows which execution it belongs to at a given time.

Notifications of an execution can only be reached; as long as the execution is not removed from the system (being terminated is allowed).

Please note that the Performance page of the launch configuration has some options to tweak the performance of this view. For more information please refer to Host Controllers Page of the Launch Configuration.

The following actions are supported:

- **Clear**:

  Selecting this action clears the notification logs of the selected execution.

- **Save as**:

  Selecting this action allows the user to save the notification message in a file via a standard **save as...** Window.

- **Follow the last record**:

  If this action is checked and new records are inserted into the lists of notifications, then the view will automatically make the last record visible. If this action is not set, then the user can stay fixed on a record (by selecting it), while new log records are still inserted.

Please note that these actions can be selected from the toolbar and menu bar of the view.

5.4. TITAN Test Results View

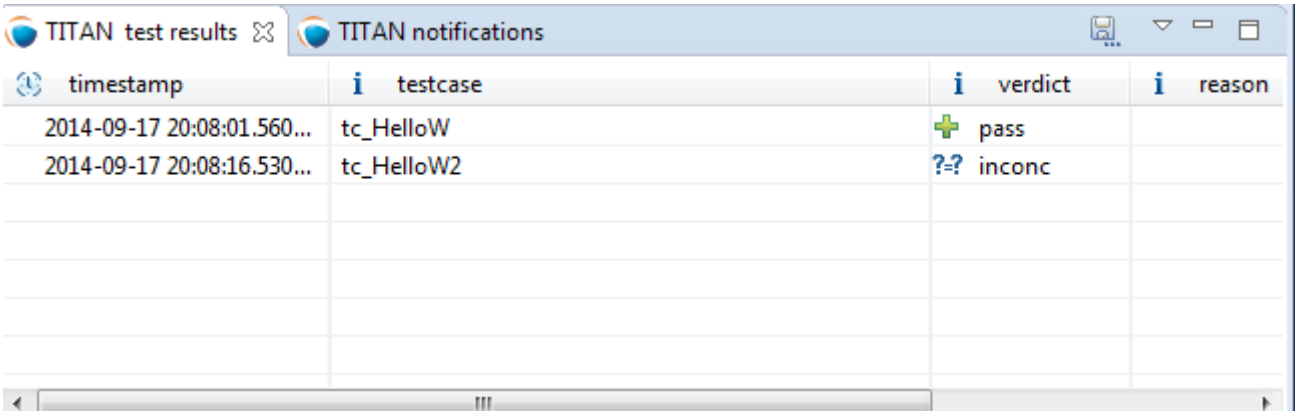


Figure 34. The Executed tests view

The TITAN test results view (above) contains verdict changing notifications extracted from the list of notifications.

This view serves the purpose of summarizing the verdict changes of a test execution, briefly showing when and which testcase set what verdict value. As several executions can be ongoing at any given time, this view always shows the verdict changing messages, created by the execution/launch actually selected in the TITAN Execution Controller view (Figure 26). This information of an execution can only be reached as long as the execution is not removed from the system (being terminated is allowed).

It is worth to notice that the tool tip of this view not only identifies the execution it belongs to, but also presents its statistics (as seen below).

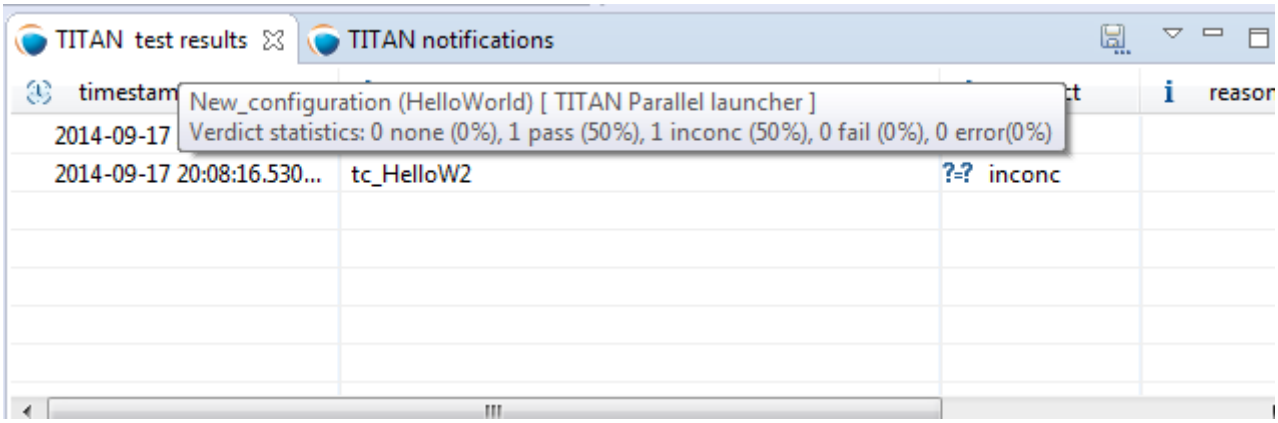


Figure 35. Executed tests view's tooltip

The following actions are supported:

- **Save as**:

  Selecting this action allows the user to save the extracted notifications in a file via a standard **save as...** Window.

- **Follow the last record**:

  If this action is checked and new records are inserted into the lists of extracted notifications, then the view will automatically make the last record visible. If this action is not set, then the user can stay fixed on a record (by selecting it), while new log records are still inserted.

Please note that these actions can be selected from the toolbar and menu bar of the view.

Please note, that if verdict extraction is not enabled on the Performance page of the launch configuration, this view will be empty. For more information please refer to section 4.2.5.

5.5. Console Views

There are a few things to remember about console views:

- For single and mctr_cli mode launching these are supported by Eclipse, for JNI mode launching the TITAN RUNTIME Console is used to output messages.
- The consoles of the single and mctr_cli mode launches and the consoles of all host controllers can be used to enter text.
- The consoles of the single and mctr_cli mode launches are executing in shells, con-figured on the Environment page of the launch configuration (please refer to section 4.2.3). The commands that execute the Main controllers in these views are prefixed with `"sh -c sleep1;"`.
  `"sh -c"` creates a new shell.
  "sleep 1" is used to have enough time to connect to the output of the started process before it actually has some outputs. (This is a technical limitation: we have experienced a few so fast executions in our tests, that at the time the output processing functionalities tried to connect to the output of the process, the process was already finished. This amount of delay is still not too much, but should already be enough to solve this kind of situations).
- When an executing launch is selected in the Executor monitor view, the Console view changes

to the console of the main controller belonging to the execution.

5.5.1. Creating a New Console View

By default the Executor perspective has only a single Console view open this can be changed by selecting the Open Console menu (Figure 36) and clicking on the New Console View option (Figure 37). By doing so a new Console view will be created in the actual perspective



Figure 36. The Open Console menu item

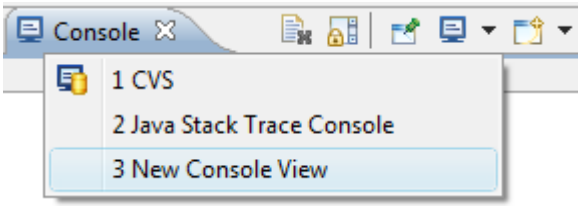


Figure 37. The New Console View menu entry

Using this feature of the Console views it is possible to create a separate Console view for each Host Controller or Main Controller of interest, and with their output at the same time.

5.5.2. Selecting and Pining to a New Console Output

The default behavior of the Console View provided by its plug-in is that it always tries to show the data printed to a console. For this reason when any Host controller or the Main Controller displays any information on its standard output, the Console View will automatically switch to the Console displaying that information.

If it is desired to have a given console present at all time this can be achieved by selecting the Display Selected Console menu item (Figure 38), and from the list of available consoles selecting the one desired (Figure 39).



Figure 38. The Display Selected Console menu

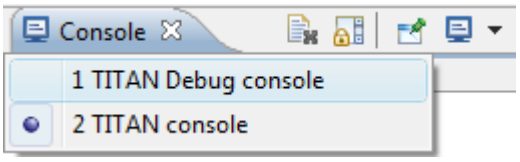


Figure 39. A list of available consoles displayed

To force a Console View to always display the contents of a given console, one has to click on the Pin Console menu item (Figure 40). To lift this limitation from the Console View the Pin Console menu item has to be clicked again.



Figure 40. The Pin Console menu item

5.6. Limitations

- In Parallel mode the actions that are displayed to be available and the ones truly available might differ in special time periods. The reason for this limitation is that it is almost impossible to tell the exact state of the Main Controller at a given time. For example, if the system buffers the output of the Main Controller, or the input of the watching process, than a short status change indicating message might not appear until the buffer is not filled up (with still to come messages). On the performance page of the mctr_cli launch configuration it can be set how often the states should be synchronized.
- In JNI mode we have witnessed, that on some machines the Java Virtual Machine fails to load the `LD_LIBRARY_PATH` environmental variable from the shell into his set of environmental variables.
  An indication of this problem is when the JNI executor reports, that the JNI dynamic library could not be loaded, though the path to the file can be found on the `LD_LIBRARY_PATH` of the shell.
  A solution can be to start Eclipse with the following command forcing the Java Virtual Machine to load this environmental variable:

```
java -classpath startup.jar Djava.library.path=$LD_LIBRARY_PATH
org.eclipse.core.launcher.Main
```

(This command must be executed in the directory where Eclipse was installed to.)

- Execution in JNI mode is not supported on Windows

Chapter 6. Other Available Functions

6.1. Formatting Log Files

To format a log file (one having `.log` as extension) **right click** the file and select **Titan Log** > **Format log**.

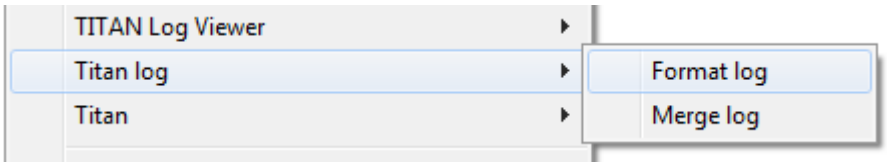


Figure 41. Format log menu

This will produce a formatted log file in the very same directory, with the same name, but having the extension `.formatted_log`.

Please note that for the duration while the formatted log is being created progress indication is provided in the **Progress view**.

6.2. Merging Log Files

To merge several log files (ones having `.log` as extension) select them, and after right clicking on one select **Titan Log** > **Merge log.**

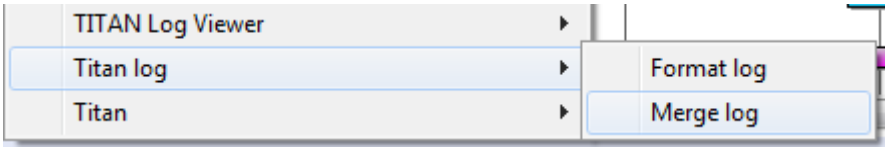


Figure 42. Merge log menu

This will first ask for the file where the results have to be saved, processing the log files will only start after a new or an existing file is selected.

Please note that for the duration while the formatted log is being created progress indication is provided in the **Progress view**.

Chapter 7. Launching TITAN Java Projects

This chapter describes the launching modes of TITAN Java projects.

After building a TITAN Java project, it is ready to be launched as a Java project. In Eclipse, every aspect of the launch can be configured, for example, different environmental settings can be created by creating different launch configurations, without modifying the system environment variables, so different test environments can be created.

NOTE	Right now the Java side of the TITAN Test Executor is supported as a Java application native to Eclipse. It as executed with the same limitations and benefits.

WARNING	The execution of TITAN Java projects (the Java side of the Test Executor) is done as Eclipse native Java applications. It is not yet fully integrated to the usual Interface elements like Views that support the execution of the binaries of the C side of the TITAN Test Executor.

7.1. The Launching Modes Supported by the TITAN Executor Plug-in for TITAN Java Projects

The TITAN Executor can operate in single or in parallel mode.

From the point of view there are 2 ways to execute TITAN Java projects: from Eclipse as Java projects and executing exported jar files.

7.1.1. Executing TITAN Java Projects from Eclipse

To execute TITAN Java projects inside Eclipse requires the creation of a Launch configuration. For TITAN Java Projects the TITAN Native Java launch configuration mode is necessary that supports both the single and parallel execution modes. Launch configurations can be created, modified and deleted in the **Create, manage, and run configuration** dialog window. It can be opened in numerous ways as detailed in Creating Launch Configuration:

- using the **Run as** option in pop-up menu of the **Project explorer** (Figure 7),
- using the **Launch Commands** on the toolbar (Figure 8).

Furthermore, a default launch configuration can be created using launch shortcuts. It works the same way as described in Creating Launch Configuration, however the TITAN Native Java launching mode requires less option, i.e.:

1. A new launch configuration is made.
2. The project and configuration file paths are initialized.
3. Finally the newly created launch configuration is launched automatically in parallel execution mode.

The launch configuration created using launch shortcuts is automatically added to favorite list

under the **Launch Commands** (Figure 8).

NOTE	It is encouraged to use the launch shortcuts to reduce the number of possible mistakes. The generated launch configuration is still editable in the **Create, manage, and run configuration** dialog window (see below).

7.1.2. Basic Main Controller Options Page of the Launch Configuration

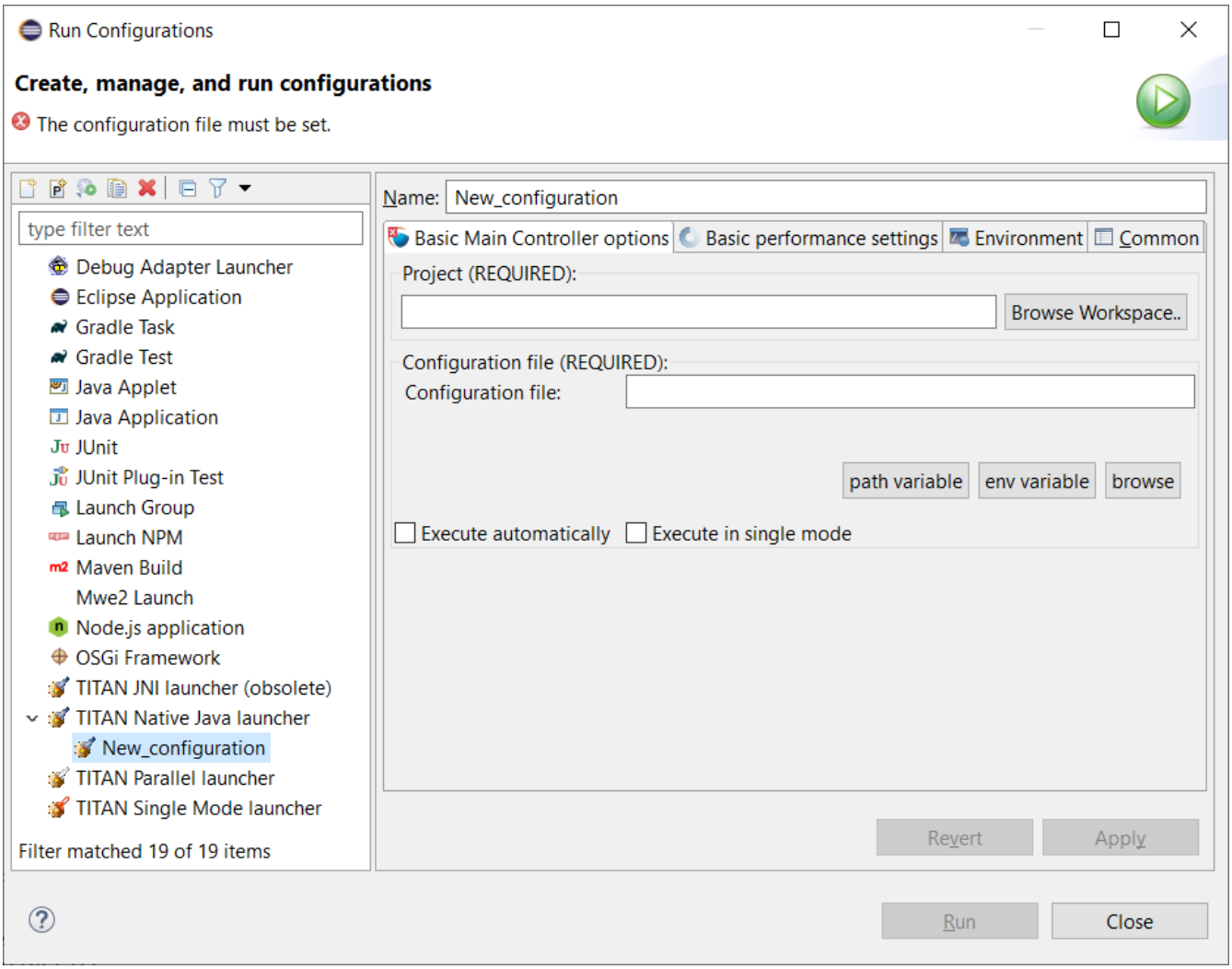


Figure 43. Basic Main Controller options page

On this page it is possible to set:

- The name of the project.

  Filling this field is mandatory. The entered name is checked for validity. The project's root folder is used during the automatically filling of the other fields. The path variables are relative to the project's root supporting the portability of the project as whole. If you enter the name of a valid project with TITAN nature (or select one by browsing, as can be seen on Figure 13), the configuration file will be filled in automatically.

NOTE	It is encouraged to use the **Browse Workspace** button to select a valid project from the workspace that simplifies the filling of the other fields, as well as reduces the possible mistakes.

- The path of the configuration file.

  Please note that not only the existence but also the validity of the configuration file is evaluated here. If a problem was found while trying to process the configuration file, the launch process will be interrupted here. Please note that this evaluation is done every time this configuration page becomes active, meaning that switching to and from this page can take some time. The entered file path is checked for validity.

- Execute automatically

  Whether the user wish to start executing the configuration file automatically when the launcher is started. Please note that this option is turned on by default.

- Execute in single mode

  Whether the user wish to start executing the TITAN Java project in single or in parallel mode. Please note that this option is turned off by default.

All fields can be filled in either by entering the proper values, or via browsing for them.

If you press **Apply** some other launch configurations will appear automatically according to the filled values in the **Create, manage, and run configuration** dialog window.

NOTE	During the filling Eclipse might ask for saving the modification. Sometimes Eclipse automatically switches to one of the additionally created launch configuration after saving the TITAN Native Java launch configuration.

The functionality of other tab pages of the TITAN Native Java launch configuration matches the ones for JNI launch mode, see Creating Launch Configuration.

7.1.3. Executing TITAN Java project via exported JAR files

This subsection describes how to export a TITAN Java project into a JAR file and how to set up the automatic JAR export for the project. Finally, the execution of the project via CLI is presented.

Exporting TITAN Java project into JAR file using wizard

It is possible to export TITAN Java project into a single JAR file and use it as executables. This feature is provided by the in-built **Runnable JAR file** export wizard of the Eclipse IDE.

To export a .jar file from a TITAN Java project:

1. Select the project in the navigator/project explorer view.
2. In the right click menu, select **Export...**.
3. In the window appearing select **Java** / **Runnable JAR file** and than **Next** (see below).

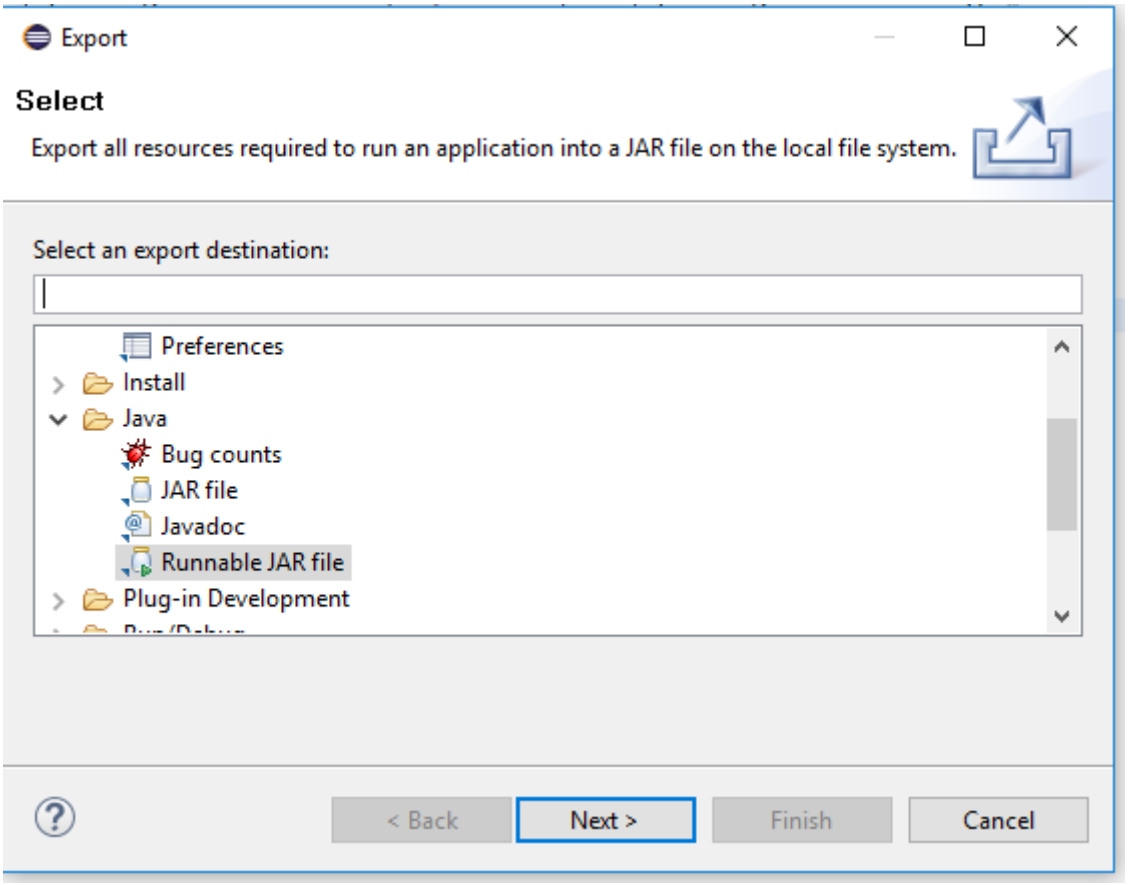


Figure 44. Export wizard

4. Configure the export (see below).

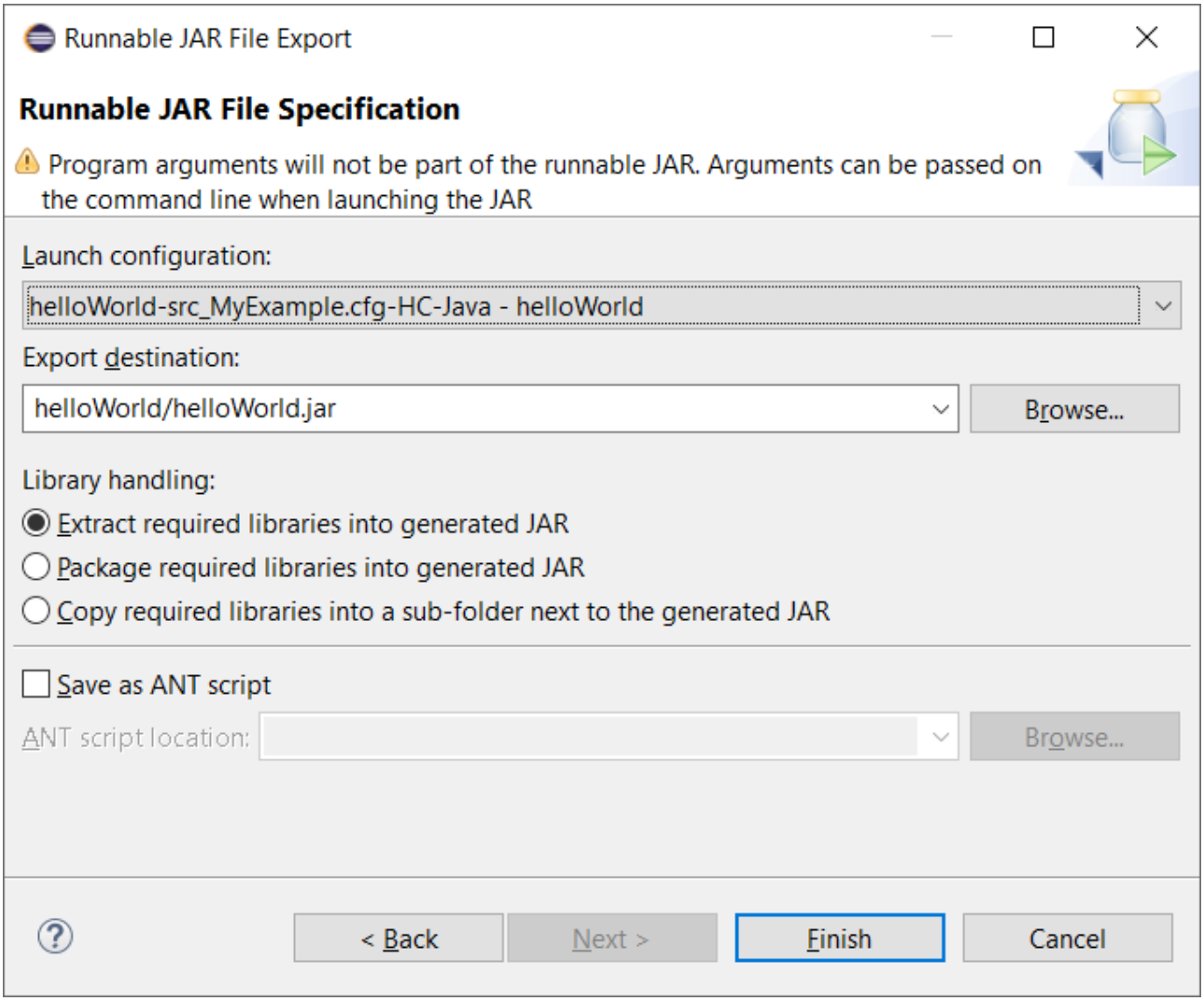


Figure 45. Export wizard for JAR file

The following options are available:

- **Launch Configuration**: select the launch configuration that configures the execution for this JAR file. This will set the class to be used for execution, i.e. the main class defined by the selected launch configuration will be the default entry point of the exported JAR file.

NOTE	Only launch configurations of Java application type are available for selection here. If such configuration is not created previously, you have to create one to continue.

- **Export destination**: select the file into which the export should be done.
- **Library handling**: It is possible to configure how the libraries are handled in the resulting jar. We recommend selecting the **Extract required libraries into generated JAR**.

5. Select **Finish**.

Automated exporting TITAN Java project into JAR file

In most cases, the manual JAR export using the Runnable JAR file export wizard is sufficient. However, in certain use case it is more convenient to automate this procedure especially when working with relatively small projects that content is often modified and JAR export is needed.

To turn on the automated JAR export, first navigate to the project properties page:

1. Select the project in the navigator/project explorer view.
2. In the right click menu, select **Properties**.

Then, the default target has to be set and the path of the JAR file has to be specified:

1. In the left tab, select **TITAN Project Property (Java)**.
2. Select **Java target creation attributes** tab.
3. Select **Executable JAR** from the drop-down list as the default target.
4. Specify the path of the JAR file in the text box.
5. Finally, apply the new settings.

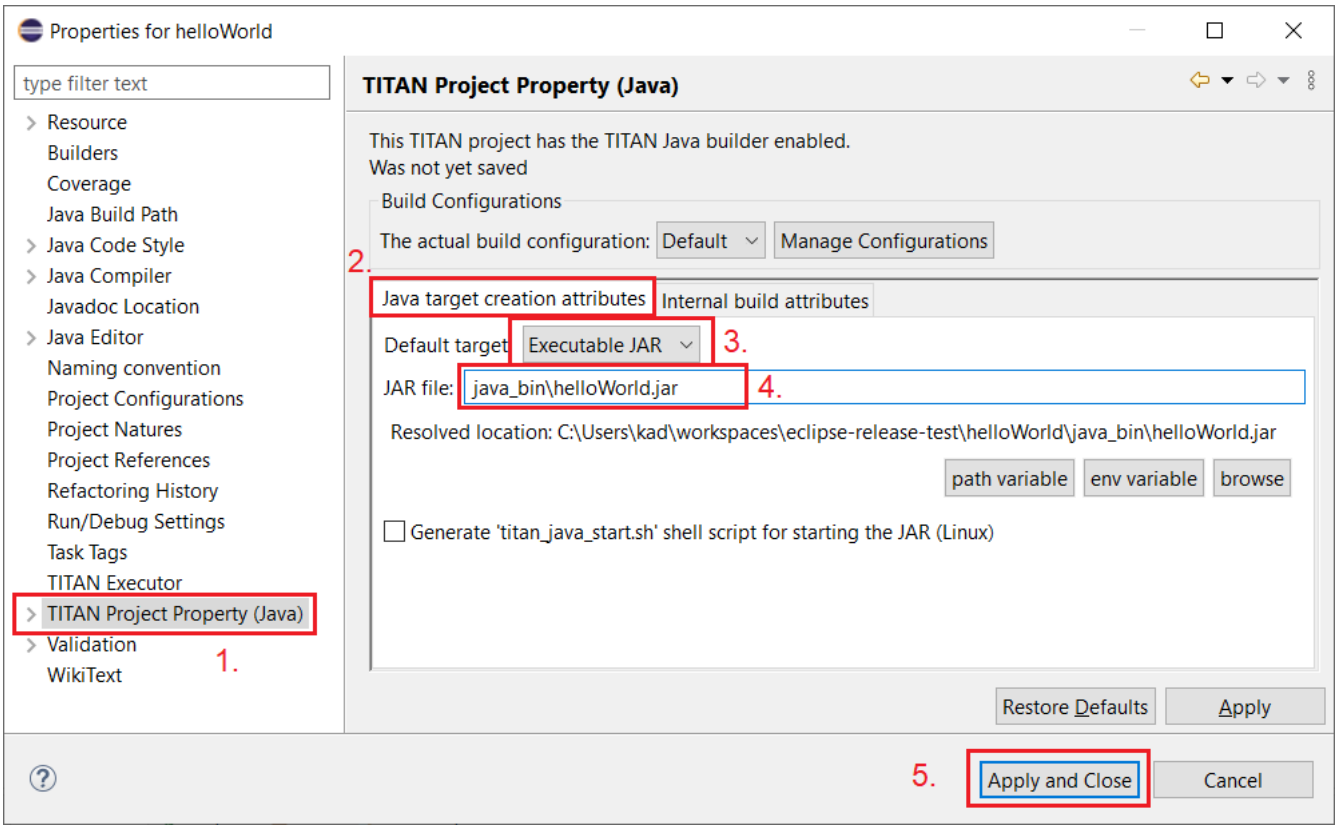


Figure 46. Turning on automated JAR export

As a result, `jarbuild.xml` ANT build script will appear in the project's root and a new ANT builder is specified for the project. To build the JAR file, the project has to be built as usual (Right click on the project/Build Project). At the end of the build process, the ANT will create the executable JAR according to the `jarbuild.xml`.

NOTE

If there is no change in the project, the build process will not start (even if the executable JAR file does not exist).

The automated JAR build procedure can be customized on demand. The generated ANT build script describes the JAR building attributes. While the newly added ANT builder automatically executes this ANT build script.

To turn off the automated JAR export, just set the default target to **Class files**.

Executing with JAR files in single mode

The Java side of the TITAN Test Executor, in the case of the exported JAR files, follows the same procedures as the C side does described in the User Guide for TITAN TTCN-3 Test Executor[3]. However, there are some differences related to executing JAR files. The executable JAR file contains the host controller for both in single and parallel mode.

For example executing a generated executable, in single mode, on the C side:

```
./helloWorld.exe config.cfg
```

Executing an exported JAR file, in single mode, on the Java side, use the following command:

```
java -jar jar_file.jar config_file.cfg
```

For example:

```
java -jar helloWorld.jar config.cfg
```

If the parallel execution mode is specified as the default entry point of the executable JAR, the following command should be used:

```
java -cp jar_file.jar org.eclipse.titan.{name of project}.generated.Single_main
config_file.cfg
```

For example:

```
java -cp helloWorld.jar org.eclipse.titan.helloWorld.generated.Single_main config.cfg
```

NOTE	During the JAR export process, the selected launch configuration determines the default entry point of the executable JAR file, i.e. either single or parallel execution mode will be started by the `java -jar` command.

Executing with JAR files in parallel mode

The Java side of the TITAN Test Executor, in the case of the exported JAR files, follows the same procedures as the C side does described in the User Guide for TITAN TTCN-3 Test Executor[3]. With differences related to executing Java files.

To execute test suites in parallel mode first the Main Controller needs to be started. It is possible to use the Main Controller provided by the TITAN TTCN-3 Test Executor:

```
$ ./mctr_cli.exe config.cfg

*********************************************************************
* TTCN-3 Test Executor - Main Controller 2                          *
* Version: 11.2.1                                                    *
* Copyright (c) 2000-2026 Ericsson Telecom AB                       *
* All rights reserved. This program and the accompanying materials      *
* are made available under the terms of the Eclipse Public License v2.0 *
* which accompanies this distribution, and is available at              *
* https://www.eclipse.org/org/documents/epl-2.0/EPL-2.0.html            *
*********************************************************************

Using configuration file: config.cfg
MC@HU-00000227: Listening on TCP port 7339.
MC2>
```

But for TITAN Java projects it is recommended to use the Java implementation of the Main Controller that is available in the exported JAR file:

```
$ java -cp jar_file.jar org.eclipse.titan.runtime.core.mctr.CliMain config.cfg

*********************************************************************
* TTCN-3 Test Executor - Main Controller 2                          *
* Version: 11.2.1                                                    *
* Copyright (c) 2000-2026 Ericsson Telecom AB                       *
* All rights reserved. This program and the accompanying materials      *
* are made available under the terms of the Eclipse Public License v2.0 *
* which accompanies this distribution, and is available at              *
* https://www.eclipse.org/org/documents/epl-2.0/EPL-2.0.html            *
*********************************************************************

Using configuration file: config.cfg
MC@HU-00000227: Listening on TCP port 7339.
MC2>
```

It will tell us, that it accepts connections on the localhost machine, on the port number 7339.

To connect to it, in parallel mode, on the C side:

```
./helloWorld.exe localhost 7339
```

Executing an exported JAR file, in parallel mode, on the Java side:

```
java -jar jar_file.jar mctr_ip mctr_port
```

For example:

```
java -jar helloWorld.jar localhost 7339
```

If the single execution mode is specified as the default entry point of the executable JAR, the following command should be used:

```
java -cp jar_file.jar org.eclipse.titan.{name of project}.generated.Parallel_main
mctr_ip mctr_port
```

For example:

```
java -cp helloWorld.jar org.eclipse.titan.helloWorld.generated.Parallel_main localhost
7339
```

NOTE

The automated JAR export feature builds JAR files in parallel execution mode by default.

Alternatively, the Java based host controller can be started using the `ttcn3_start` script in parallel mode. It works the same way as starting an executable generated by the C side of the TITAN Test Executor. For example:

```
./ttcn3_start helloWorld.jar config.cfg
```

Tips

It is possible to provide Java VM arguments when executing exported JAR files. For example:

```
java -Xmx1024m -jar helloWorld.jar config.cfg
```

Executing with `titan_java_start` script

Upon configuring the automated JAR export in the **TITAN Project Property (Java)** page, it is possible to generate a shell script called `titan_java_start.sh` to simplify the start, as well as to support the integration of JAR-based execution in CLI environment (see Figure 46). The name of this script is not configurable and it is created beside the JAR file on request. The `titan_java_start.sh` script is generated with default values according to the project, however these default values can be be overwritten by using command line options. Therefore, the script can be used to start arbitrary executable JAR package of TITAN Java project. The following arguments and options are supported by the script:

- `configuration_file.cfg`

  It is **mandatory** to specify a configuration file optionally with its path. The accepted file extension is `.cfg`.

- `executable_JAR.jar`

  By default the exported JAR file in the same directory is used for this value. It can be overwritten by simply specifying another executable JAR file optionally with its path. This might be necessary, if the script and the JAR file are not located in the same directory. The accepted file extensions for executable JAR files are `.jar` or `.zip`.

- `-m|--mode [hc|host_controller|mc|main_controller|p|parallel|s|single]`

  By this option the execution mode can be configured. By default the automatic execution in parallel mode is used by utilizing the `ttcn3_start` script, thus installing and configuring the TITAN C/C++ Compiler and Test Executor is required for this mode. Further options are: **parallel execution mode (default):** **`p|parallel`** single execution mode: `s|single` **host controller only:** **`hc|host_controller`** main controller only: `mc|main_controller`

  + The latter two option might be necessary, if a distributed and/or parallel test execution is desired. In this case the same JAR file can be populated and used on separate machines along with the same `titan_java_start.sh` script.

- `-p|--project project_name`

  By this option the project name contained by the executable JAR file can be overwritten (see section Executing with JAR files in single mode and Executing with JAR files in parallel mode). The default project name is determined during the generation of the `titan_java_start.sh` script (see Figure 46).

The order of the listed arguments and options is arbitrary.

Chapter 8. References

- [1] Installation guide for TITAN TTCN-3 Test Executor
- [2] Installation Guide for TITAN Designer and TITAN Executor for the Eclipse IDE
- [3] User Guide for TITAN TTCN-3 Test Executor
- [4] Programmers Technical Reference for TITAN TTCN-3 Test Executor
- [5] API Technical Reference for TITAN TTCN-3 Test Executor
- [6] Release Notes for TITAN TTCN-3 Test Executor
- [7] TTCN–3 Style Guide
- [8] Methods for Testing and Specification (MTS);The Testing and Test Control Notation version 3.Part 1: Core Language European Telecommunications Standards Institute. ES 201 873-1 Version 4.5.1, April 2013
- [9] Methods for Testing and Specification (MTS);The Testing and Test Control Notation version 3.Part 4: TTCN–3 Operational Semantics European Telecommunications Standards Institute. ES 201 873-4 Version 4.4.1, April 2012