What’s New in HCL RTist 11.1

updated for release 2021.24

HCL SOFTWARE

Overview

- RTist 11.1 is based on Eclipse 2020.06 (4.16)
- HCL RTist is 100% compatible with IBM RSARTE. All features in IBM RSARTE are also present in HCL RTist. However, HCL RTist contains a few features that do not exist in IBM RSARTE.
  - Those features are marked in this presentation by **RTist only**

HCL RTist
Version: 11.1.0.v20210618_1548
Release: 2021.24

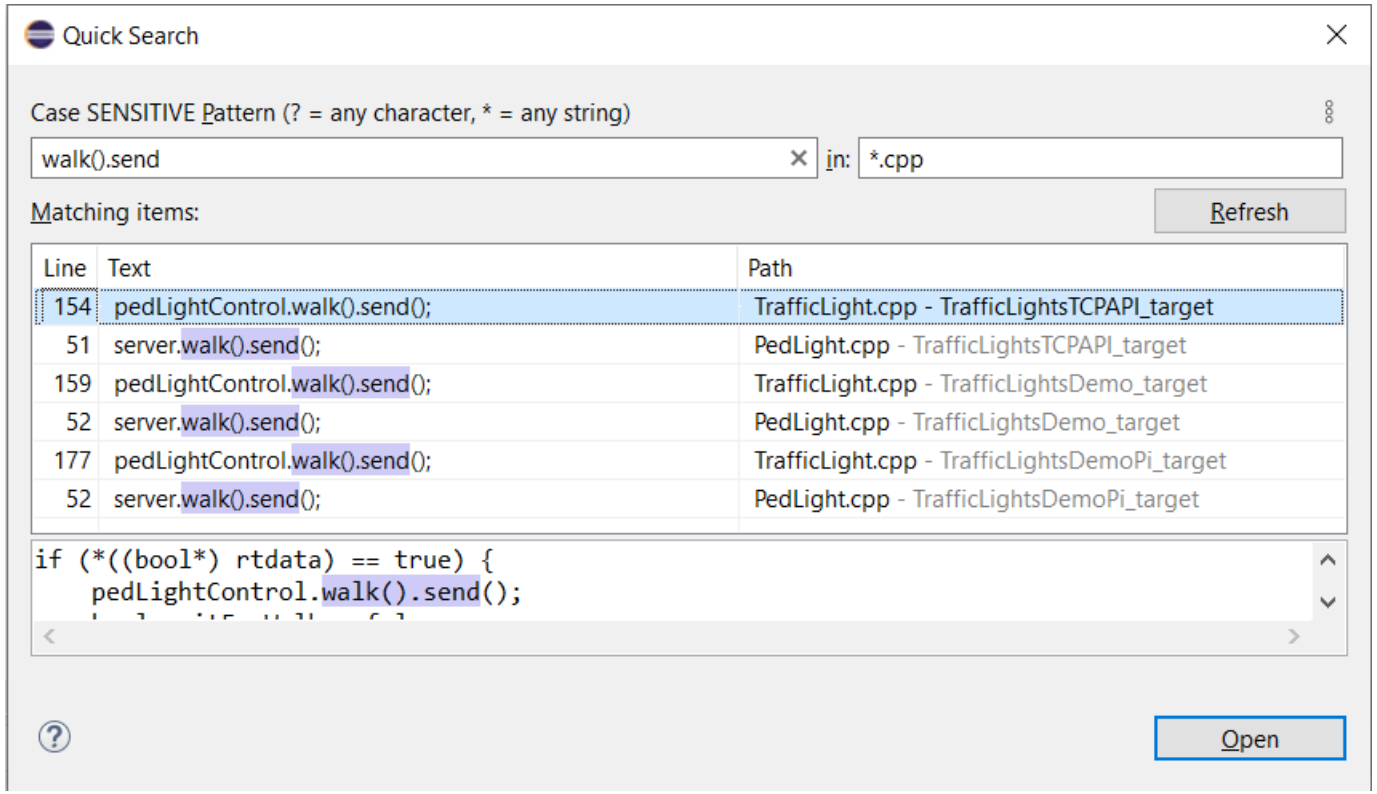
(c) Copyright IBM Corporation 2004, 2016. All rights reserved.
(c) Copyright HCL Technologies Ltd. 2016, 2021. All rights reserved.
Visit https://RTist.hcldoc.com/help/topic/com.ibm.xtools.rsarte.webdoc/users-guide/overview.html

Eclipse 4.16 (2020.06)

- Compared to RTist 11.0, RTist 11.1 includes new features from 4 quarterly Eclipse releases:
  - 2019.09 (https://www.eclipse.org/eclipse/news/4.13/platform.php)
  - 2019.12 (https://www.eclipse.org/eclipse/news/4.14/platform.php)
  - 2020.03 (https://www.eclipse.org/eclipse/news/4.15/platform.php)
  - 2020.06 (https://www.eclipse.org/eclipse/news/4.16/platform.php)
- For full information about all improvements and changes in these Eclipse releases see the links above
  - Some highlights are listed in the next few slides...

Eclipse 4.16 (2020.06)

- A new Quick Search dialog allows you to search the files of your workspace faster (“as-you-type”)
  - For a similar search experience in model files, use the Find Named Element command instead

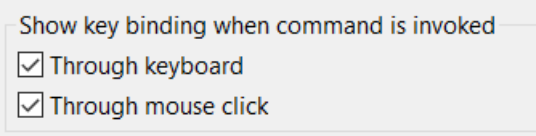


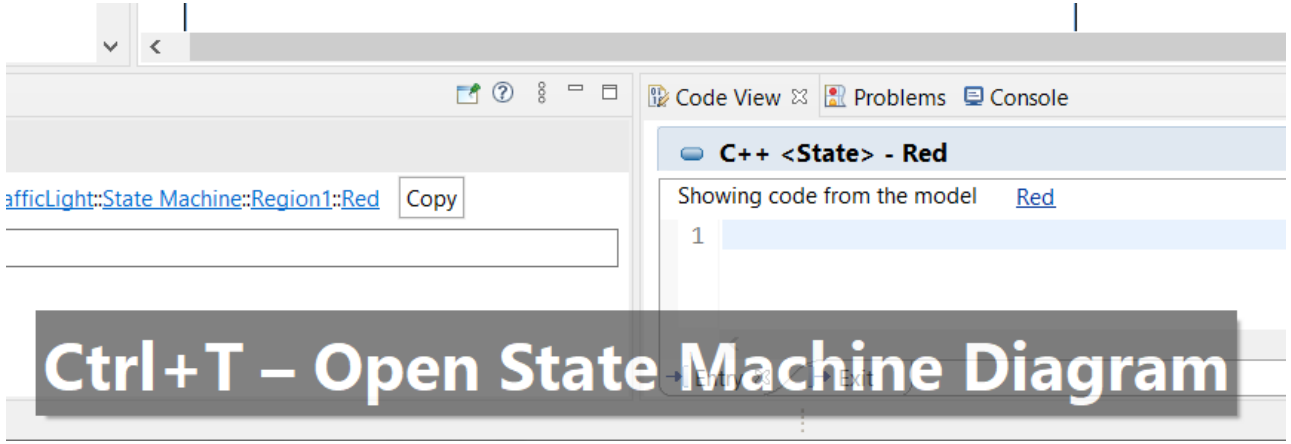
HCL SOFTWARE

Eclipse 4.16 (2020.06)

- By default at most 99 editors can now be open at the same time
  - Helps keeping the performance good when working with Eclipse for a long time
  - This can be controlled by the preference **General – Editors – Close editors automatically**

- Showing key bindings when performing commands
  - New preferences in **General – Keys**
  - This is a good way to learn about key bindings for the commands that are used, and can also help in presentations

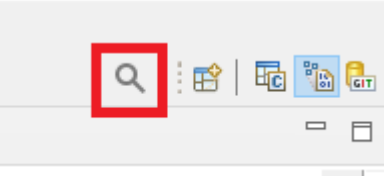


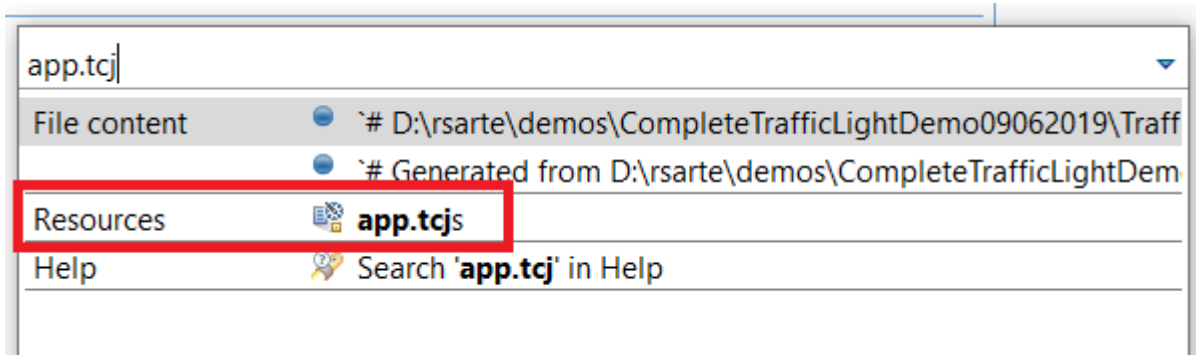


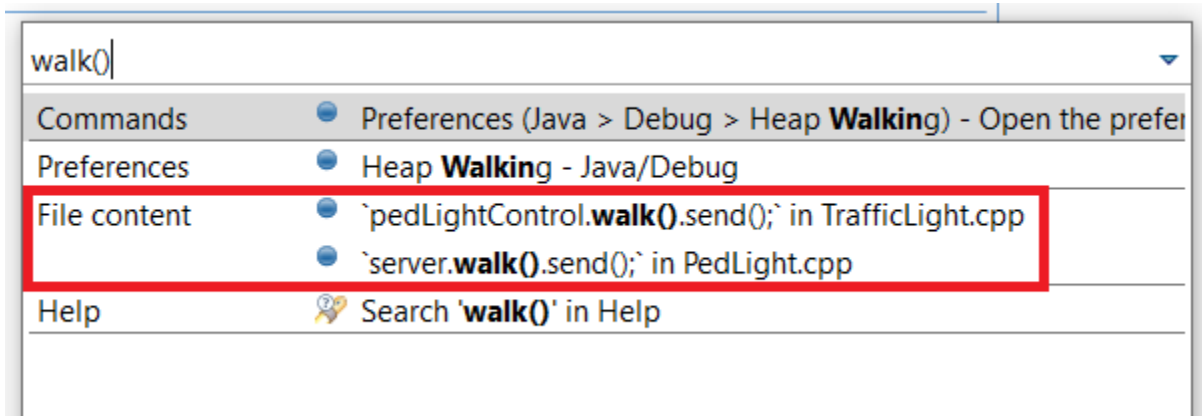
HCL SOFTWARE

Eclipse 4.16 (2020.06)

- Quick Access field replaced with toolbar button
  - Takes less space in the toolbar, and instead uses a normal dialog for typing an showing the results
  - Same key binding as before (Ctrl + 3) but the command is now called “Find Actions”
  - The results now also include matching files in the workspace, and text matches in files (requires that Quick Search has been used at least once)







HCL SOFTWARE

Eclipse 4.16 (2020.06)

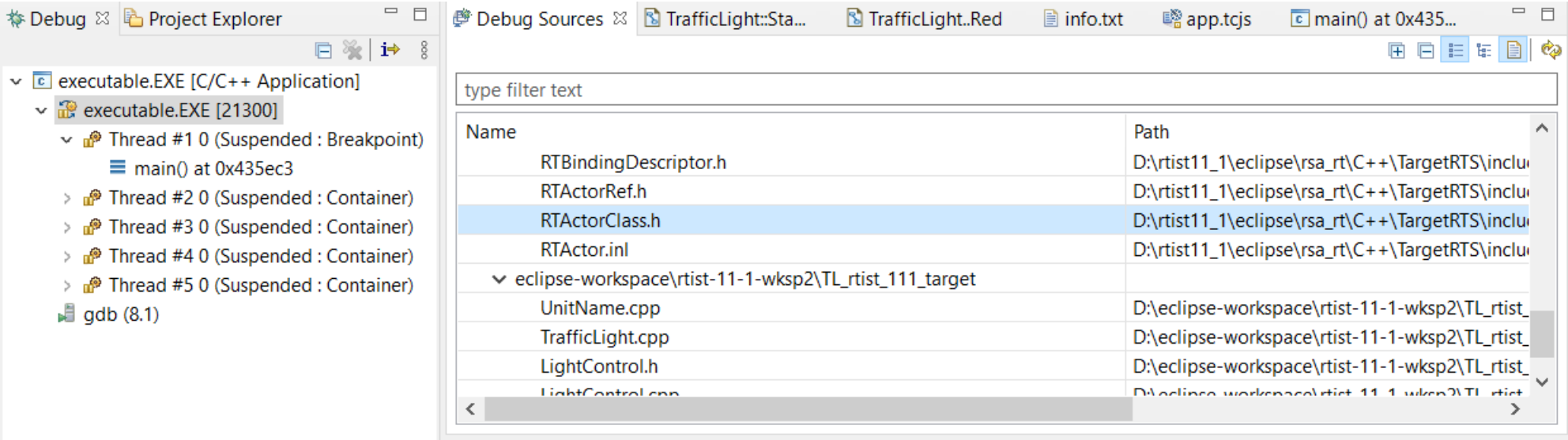
- Show code problems inline
  - Makes errors/warnings more visible and lets you apply quick fixes without having to go to the Problems view
  - Enable this feature in preferences at **General – Editors – Text Editors – Show code minings for problem annotations**

```
34 {
35 //{{{USR platform:/resource/TL_rtist_111/Tr
36 log.log("Red -> Green");
   'cout' is not a member of 'std'
37 std::cout << "test";
38 //}}}USR
39 }
```

- There were several improvements in SWT and GTK
  - The minimal supported GTK version is now 3.20

CDT 9.11 (included as part of Eclipse 2020.06)

- New Debug Sources view
  - Shows source files the C++ debugger knows about when debugging an application
  - Useful in particular when the application contains source files that are not present in the Eclipse workspace
  - Source files can be found by searching (filtering) and opened by double-click



HCL SOFTWARE

CDT 9.11 (included as part of Eclipse 2020.06)

- CODAN improvements
  - Several additional checks implemented

- For more information about CDT improvements see
  https://wiki.eclipse.org/CDT/User/NewIn99
  https://wiki.eclipse.org/CDT/User/NewIn910
  https://wiki.eclipse.org/CDT/User/NewIn911

Newer EGit Version in the EGit Integration

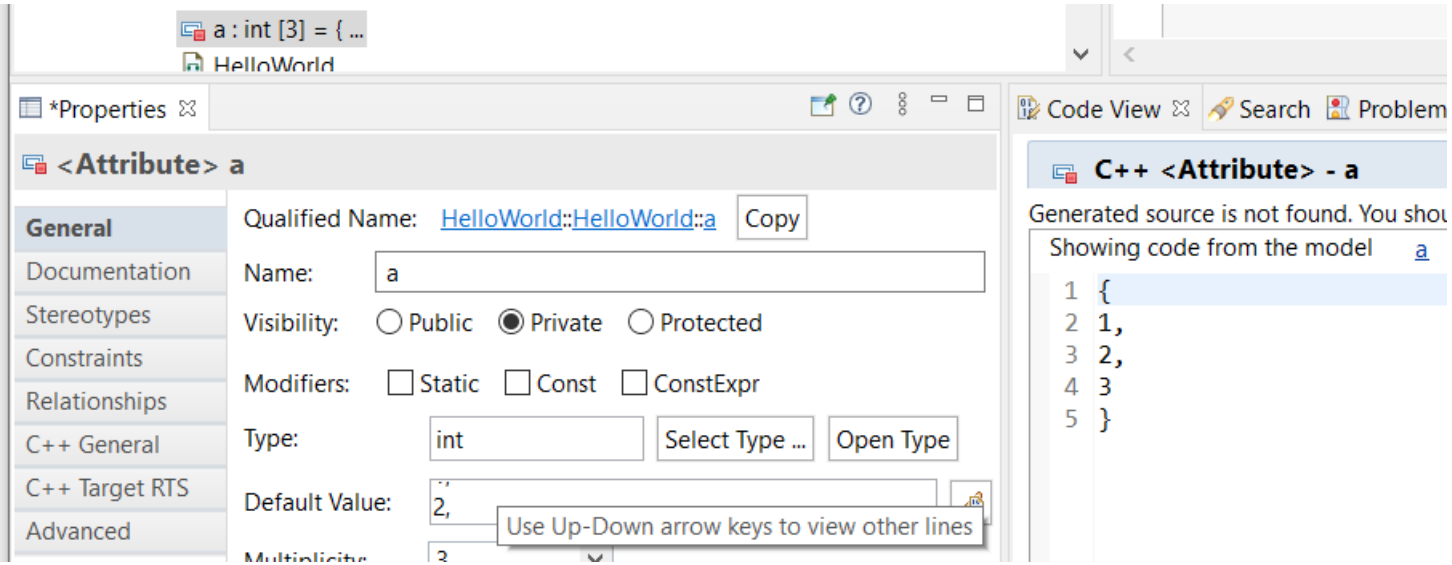
- The EGit integration in RTist has upgraded EGit from 5.4 to 5.8
  - This is the recommended and latest version for Eclipse 2020.06
- This upgrade provides several new features, performance improvements and bug fixes
  - For detailed information about the changes see
    https://wiki.eclipse.org/EGit/New_and_Noteworthy/5.5
    https://wiki.eclipse.org/EGit/New_and_Noteworthy/5.6
    https://wiki.eclipse.org/EGit/New_and_Noteworthy/5.7
    https://wiki.eclipse.org/EGit/New_and_Noteworthy/5.8

Installation Script

- A bash script is now available which helps automating the installation of RTist
  - Download it from the Info Center
  - Works on both Windows and Linux
- In particular useful for installing RTist 11.1 (due to the requirement of using Java 11 for the installation)
  - Choose whether you want to then run RTist with either Java 8 or Java 11
- For documentation on how to configure and use the script see the Info Center.

Properties View Improvements

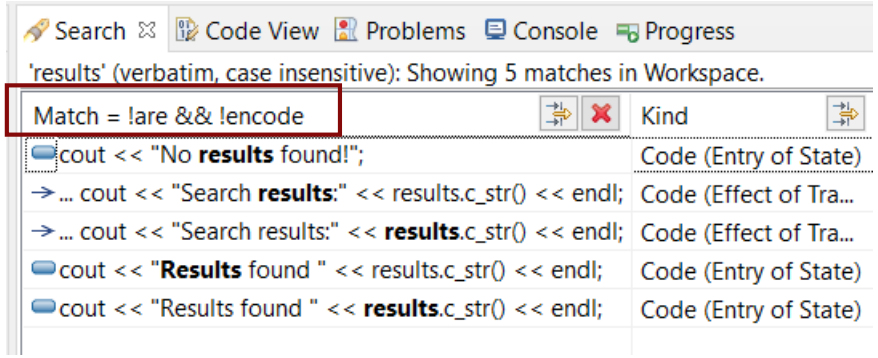
- The Default Value field now supports multi-line values
  - To create a multi-line default value you still need to use the Code View or Code Editor
  - For editing a multi-line default value you can now use the Properties view, but it's still often more convenient with the Code View or Code Editor
  - For quickly viewing a multi-line default value the Properties view can be handy



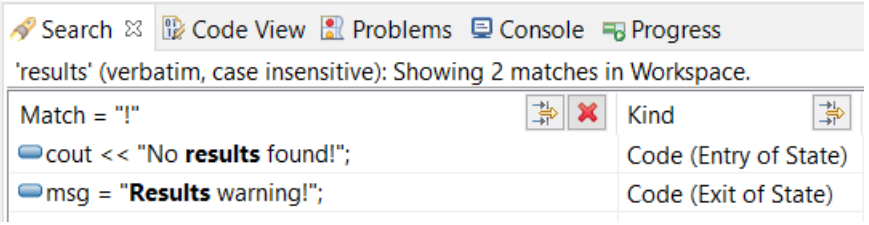
HCL SOFTWARE

Search Filtering

- It's now possible to filter search results using Boolean operators NOT (!) and AND (&&)
  - Useful if a search returns too many matches
  - Use a filter on the form
    `!A && !B && ... !X` to hide matches where certain words <u>are not</u> present
  - Use a filter on the form
    `A && B && ... X` to only show matches where certain words <u>are</u> present
  - ...or any combination, where some words are present and others not



- Enclose the filter string in double quotes to apply the filter verbatimly
  - Needed if the filter string contains the characters `!` or `&&`



Generic Type Descriptors

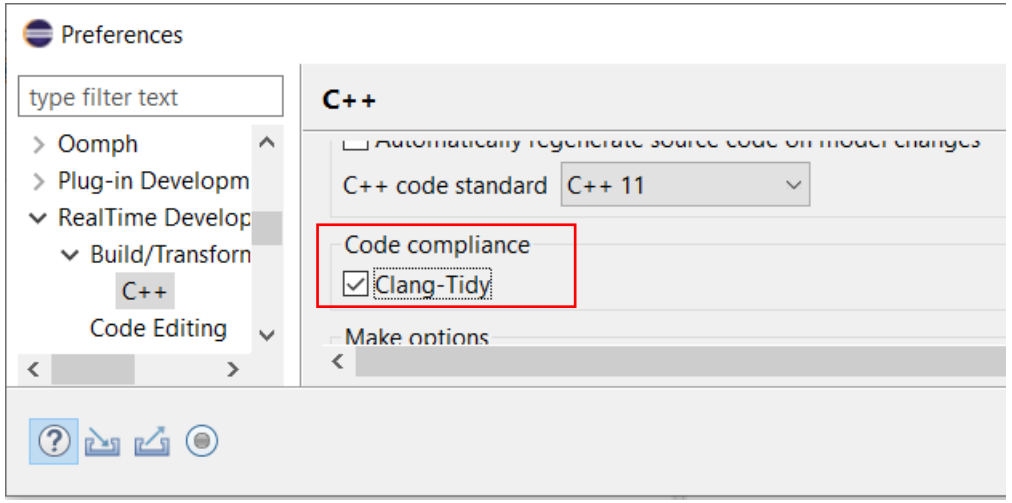
- The model compiler now supports generating type descriptors for type aliases with template parameters
  - For example: `template<typename T, unsigned int N > using StdArray = std::array<T, N>;`
  - If type descriptor functions are defined for the type alias, they will be generated as template functions with the same template parameters
  - Allows to implement generic type descriptors that work for all (or many) instantiations of the template
  - A new `RTObject_class::fromType<T>()` template function can be used for looking up the type descriptor of a type at compile time. Useful for example when implementing generic encode or decode functions. Specialize it for the types that you use (specializations for built-in types are available in the TargetRTS). For example:

```
template <> inline const RTObject_class* RTObject_class::fromType<RTString>() {
    return &RTType_RTString;
}
```

- You can specify a unique name for the type descriptor of a specific template instantiation
  - For example: `template <> const char* RTName_StdArray<StdString, 4>::name = "StdArray<StdString, 4>";`
  - The TargetRTS now prints a warning if two type descriptors with the same name exists. Helps troubleshooting missing template specializations for the name attribute.

Code Compliance

- A new group of preferences were introduced to let the model compiler generate code according to certain code compliance rules
  - As a first step support for one specific Clang-Tidy rule is implemented
  - It suppresses warnings for use of static_cast to downcast event data in transition functions



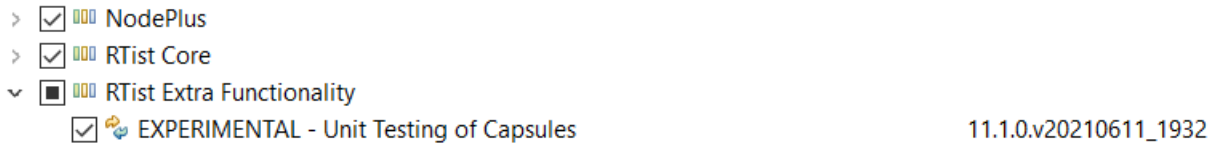

```
transition2_t1( static_cast< const bool * > ( msg->data ), static_cast< P::Base * > ( msg->sap()
/* NOLINT(cppcoreguidelines-pro-type-static-cast-downcast) */ ) );
```

Unit Testing of Capsules using the Mocha Framework (1/3)

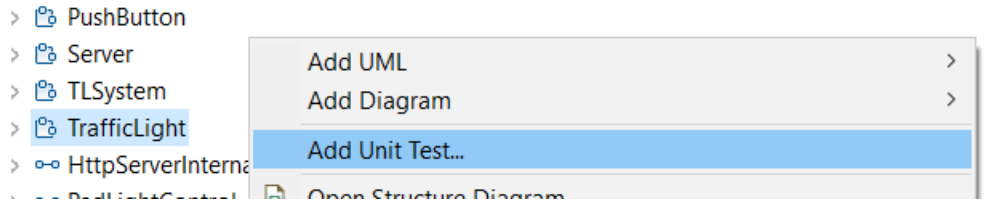
RTist only

- Mocha is a popular JavaScript framework for testing asynchronous applications
- It's now possible to use Mocha also for unit testing capsules
  - Provided by a new component that can be selected when installing
  - Note that it depends on NodePlus and is currently an EXPERIMENTAL feature

simple, flexible, fun



- To create a Mocha unit test for a capsule, invoke the new context menu command **Add Unit Test**

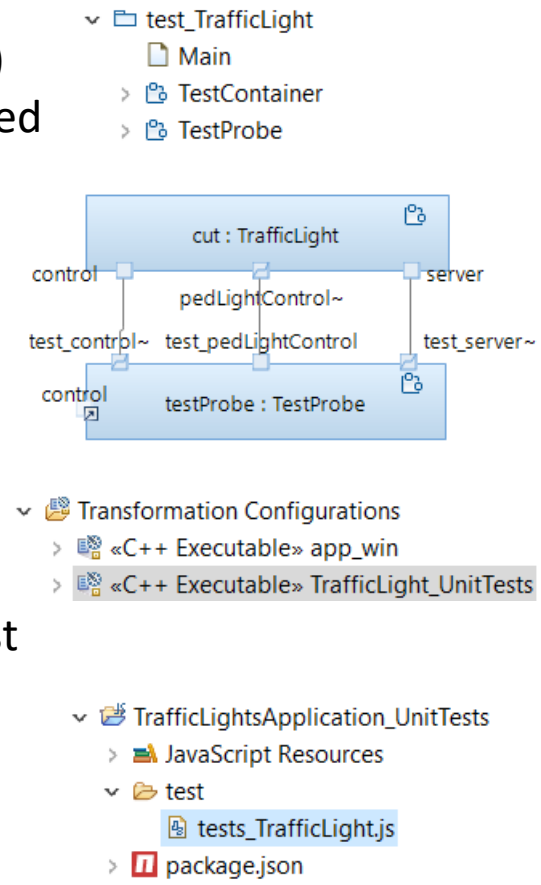


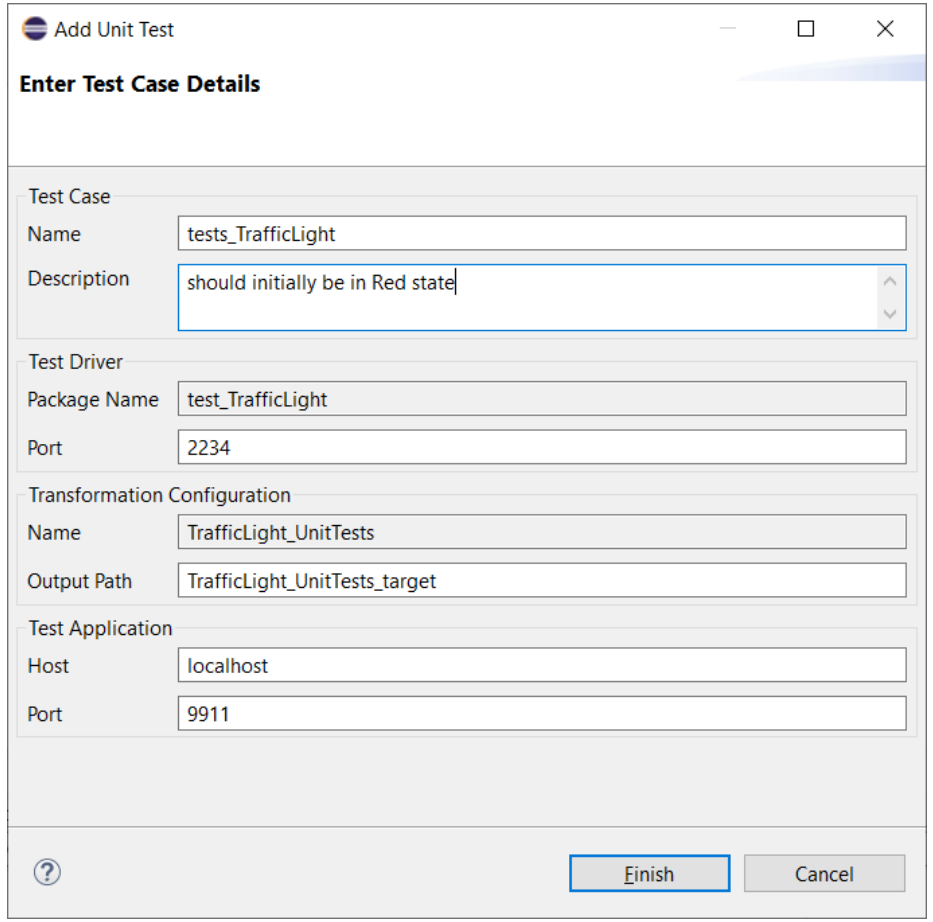
HCL SOFTWARE

Unit Testing of Capsules using the Mocha Framework (2/3)

RTist only

- The **Add Unit Test** command creates everything necessary for writing a unit test for the capsule
  - A test driver model where all service ports of the capsule under test (”cut”) are connected to similar but conjugated ports of a test probe capsule
  - A TC for building the test driver model into an executable that uses the TcpServer library for exposing all test probe ports to the Mocha test script
  - A Node.js project with a Mocha test script ready to implement the unit test





HCL SOFTWARE

Unit Testing of Capsules using the Mocha Framework (3/3)

RTist only

- The unit test can be executed right away
  - Build the test driver TC (only needed the first time, and whenever you change the capsule under test)
  - Install the Node.js dependencies for the JavaScript project (right-click on the project and do **Run As – npm install** (only needed the first time – it is assume you already have installed Mocha on the machine)
  - Run the testcase by right-click on the .js file and do **Run As – JavaScript Unit Test**

tests_TrafficLight.js | TestContainer | TrafficLight_UnitTests.tcjs

```javascript
var assert = require('assert');
describe('TrafficLight', function() {
    it('should initially be in Red state', function() {
        this.timeout(15000);
        const testProbe = require ('rt-test-probe')('localhost', 9911);
        return testProbe.startListenForEvents(2234)
        .then((data) => {
            // TODO: Implement test here
        })
        .finally(() => {
            testProbe.stopListenForEvents();
        });
    });
});
```

- The test execution result is shown in the **JavaScript Unit Test** view

JavaScript Unit Test | Code View | Problems | **Console** | Error Log | Terminal

Runs: 1/1 Errors: 0 Failures: 0

TrafficLight
should initially be in Red state (0,026 s)

Failure Trace

HCL SOFTWARE

HCL

Relationship™

BEYOND THE CONTRACT

$7 BILLION ENTERPRISE | **110,000** IDEAPRENEURS | **31** COUNTRIES

WATCH THE FILM