


What's New in HCL RTist 11.2

updated for release 2022.26

Overview

- ▶ RTist 11.2 is based on Eclipse 2021.06 (4.20)
- ▶ HCL RTist is 100% compatible with IBM RSARTE. All features in IBM RSARTE are also present in HCL RTist. However, HCL RTist contains some features that do not exist in IBM RSARTE.
 - Those features are marked in this presentation by



 HCL RTist
Version: 11.2.0.v20220530_0853
Release: 2022.21

(c) Copyright IBM Corporation 2004, 2016. All rights reserved.

(c) Copyright HCL Technologies Ltd. 2016, 2022. All rights reserved.

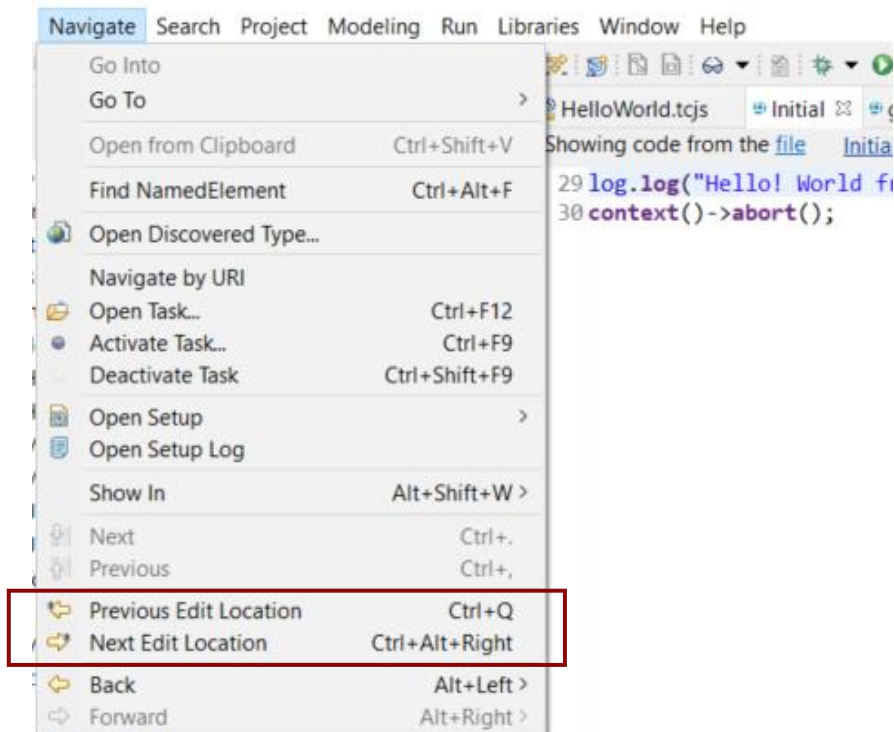
Visit <https://RTist.hcldoc.com/help/topic/com.ibm.xtools.rsarte.webdoc/users-guide/overview.html>

Eclipse 4.20 (2021.06)

- ▶ Compared to RTist 11.1, RTist 11.2 includes new features and bug fixes from 4 quarterly Eclipse releases:
 - 2020.09 (<https://www.eclipse.org/eclipse/news/4.17/platform.php>)
 - 2020.12 (<https://www.eclipse.org/eclipse/news/4.18/platform.php>)
 - 2021.03 (<https://www.eclipse.org/eclipse/news/4.19/platform.php>)
 - 2021.06 (<https://www.eclipse.org/eclipse/news/4.20/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...

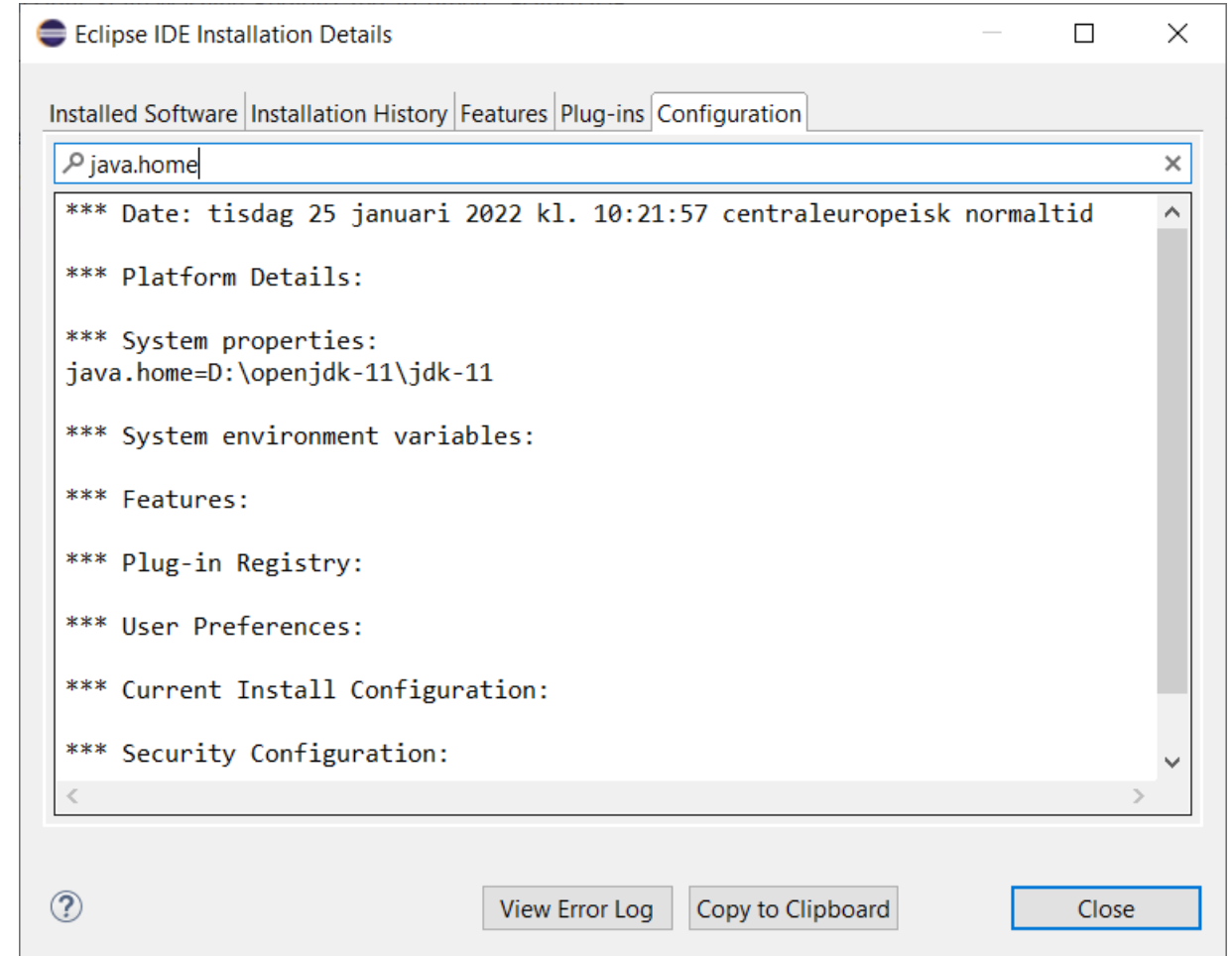
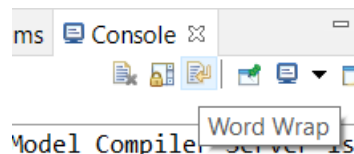
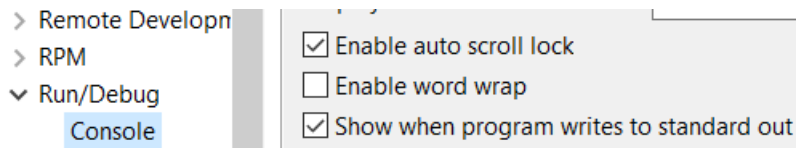
Eclipse 4.20 (2021.06)

- ▶ The **Last Edit Location** command was improved to support a list of previous edit locations
 - Now two commands are available for moving backwards and forwards in the history of recent edit locations
 - **Previous Edit Location** (Ctrl+Alt+Left Arrow or Ctrl+Q) → moves backward in the history
 - **Next Edit Location** (Ctrl+Alt+Right Arrow) → moves forward in the history
- ▶ These commands work for all Eclipse text editors (including the Code Editor but excluding diagram editors)



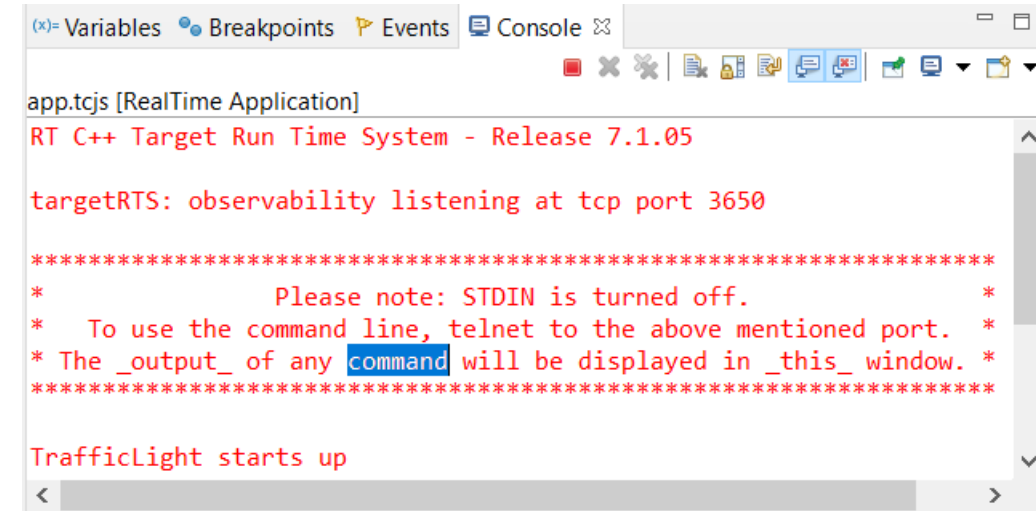
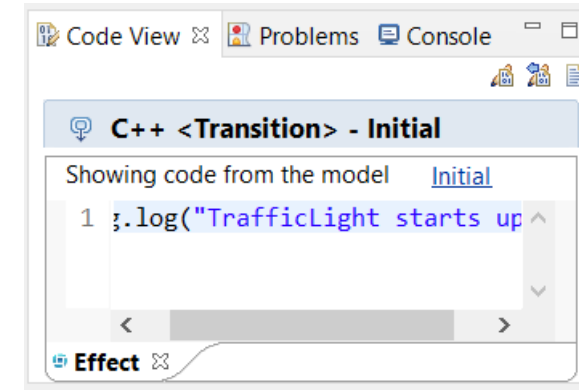
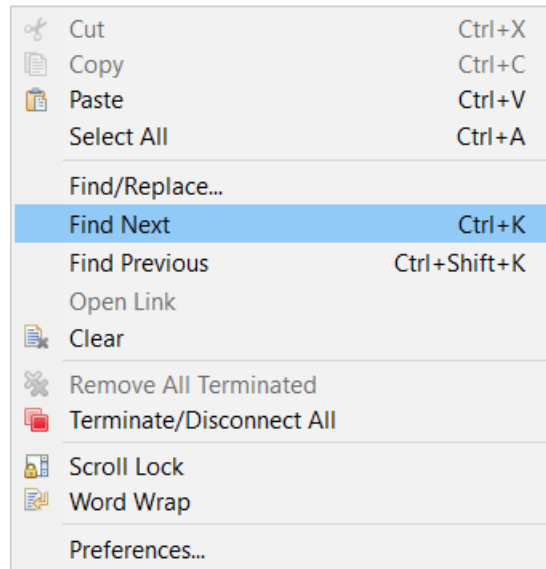
Eclipse 4.20 (2021.06)

- ▶ Filter field for the Configuration view of the Installation Details dialog
 - Makes it much faster to find particular interesting information from the configuration information (e.g. which RTist installation or JVM is being used)
- ▶ Word wrap in Console view is now saved between Eclipse sessions
 - A new preference **Run/Debug – Console – Enable word wrap** remembers this setting



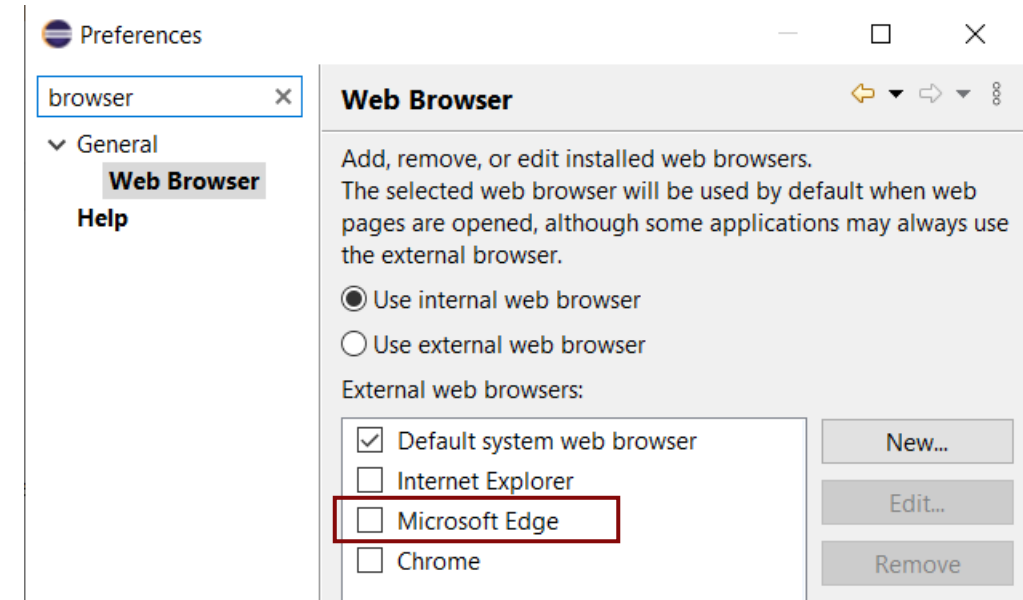
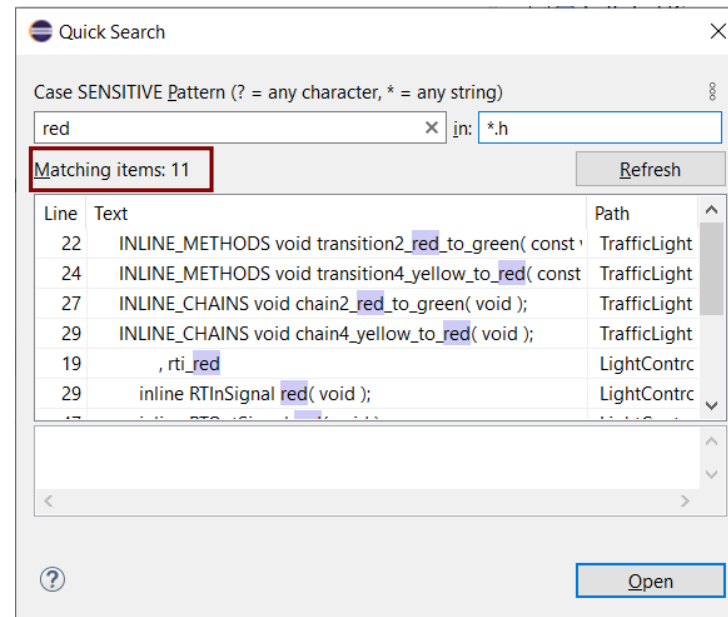
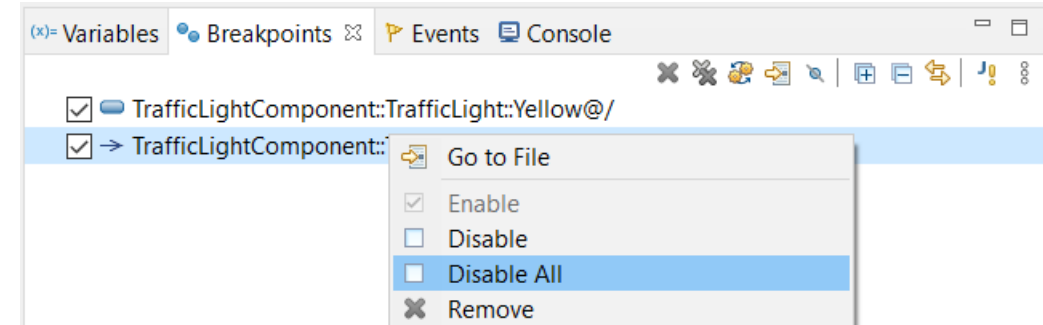
Eclipse 4.20 (2021.06)

- ▶ Horizontal scrolling with Shift + mouse wheel
 - More convenient way of scrolling horizontally if you use a mouse with a scroll wheel
 - Works in all editors (both text editors and diagram editors) and also in views (e.g. the Code view)
- ▶ Easier to repeat a search in the Console view
 - Incremental search (Ctrl+J) does not work in the Console view
 - But now you can instead use new context menu commands **Find Next** and **Find Previous** for repeating a search that was previously done with Ctrl + F.



Eclipse 4.20 (2021.06)

- ▶ Disable all breakpoints
 - A new context menu command in the Breakpoints view makes this easier
- ▶ Microsoft Edge is now supported as an external web browser
- ▶ The Quick Search dialog now shows the number of matching items
 - A new preference **General – Quick Search – Max Results** allow to stop the search when a certain number of matches have been found



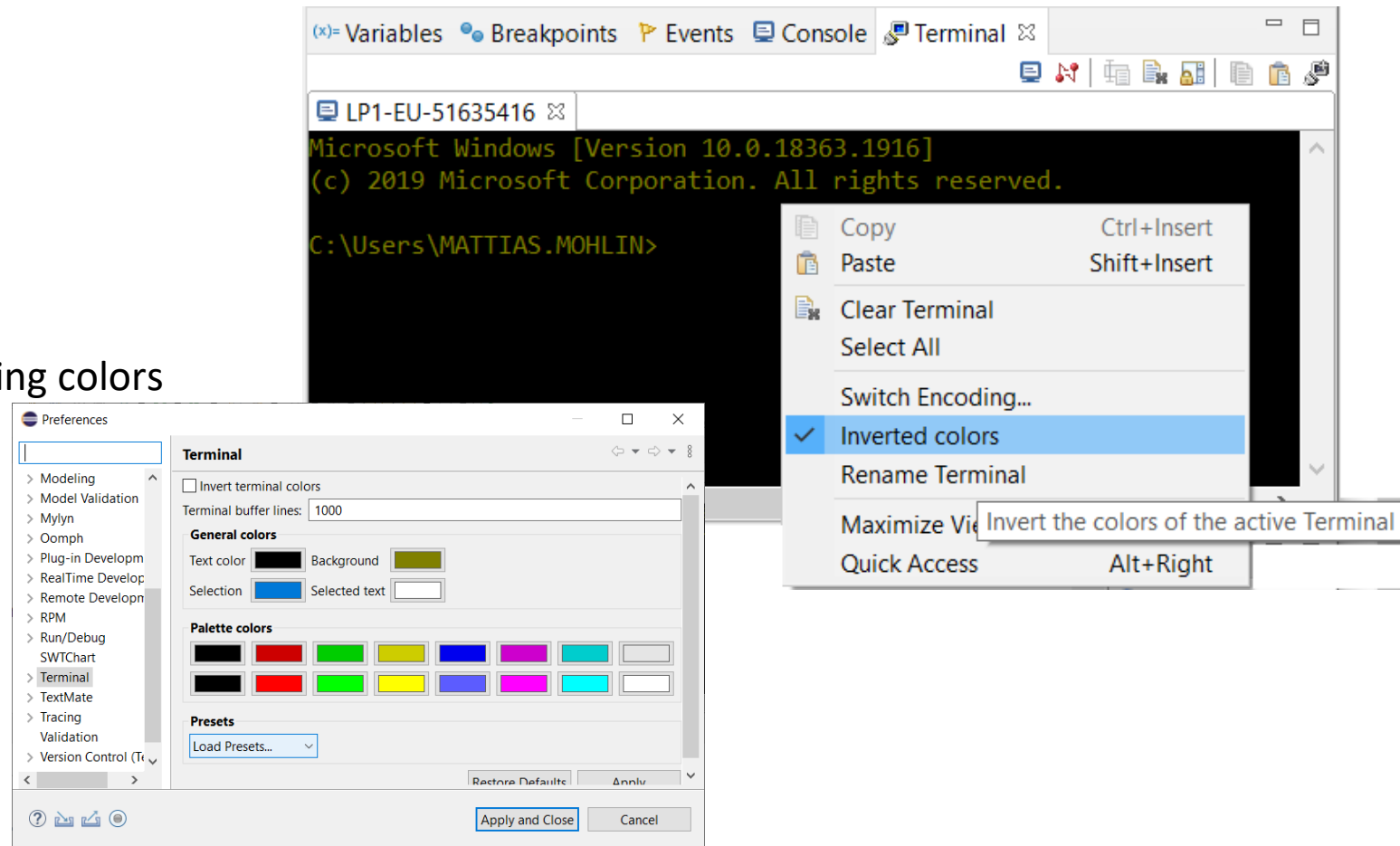
CDT 10.3 (included as part of Eclipse 2021.06)

▶ Various parser and preprocessor improvements for new C++ constructs

- Template deduction guides (C++ 17)
- `__has_include` (C++ 17)

▶ More configurable Terminal view

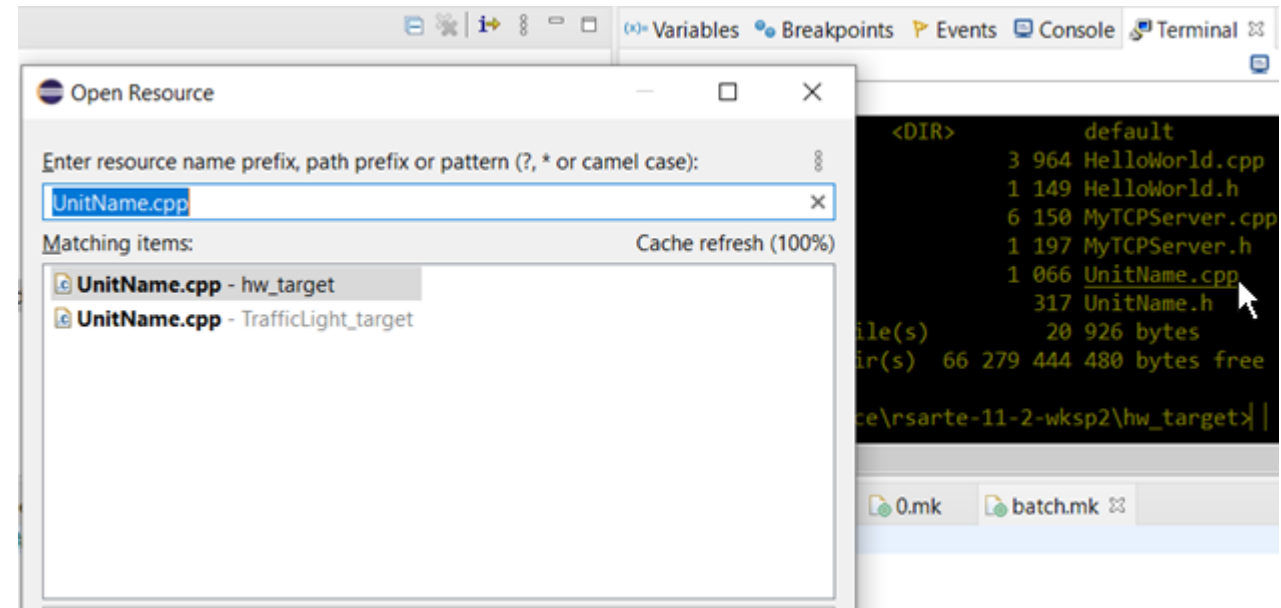
- New context menu command for inverting colors
- New preferences for configuring the colors used by the Terminal view
- Changing the colors helps for example when connecting to certain remote systems that make assumptions about what colors are used
- New context menu command for renaming the terminal (useful if you have many open at the same time)



CDT 10.3 (included as part of Eclipse 2021.06)

- ▶ Open files and links from the Terminal view
 - Ctrl+click on files shown in the Terminal view now opens the file in the workspace (sometimes via the Open Resource dialog to resolve ambiguities)
 - Ctrl+click on hyperlinks to open them in a web browser

- ▶ For more information about CDT improvements see
 - <https://wiki.eclipse.org/CDT/User/NewIn100>
 - <https://wiki.eclipse.org/CDT/User/NewIn101>
 - <https://wiki.eclipse.org/CDT/User/NewIn102>
 - <https://wiki.eclipse.org/CDT/User/NewIn103>



Newer EGit Version in the EGit Integration

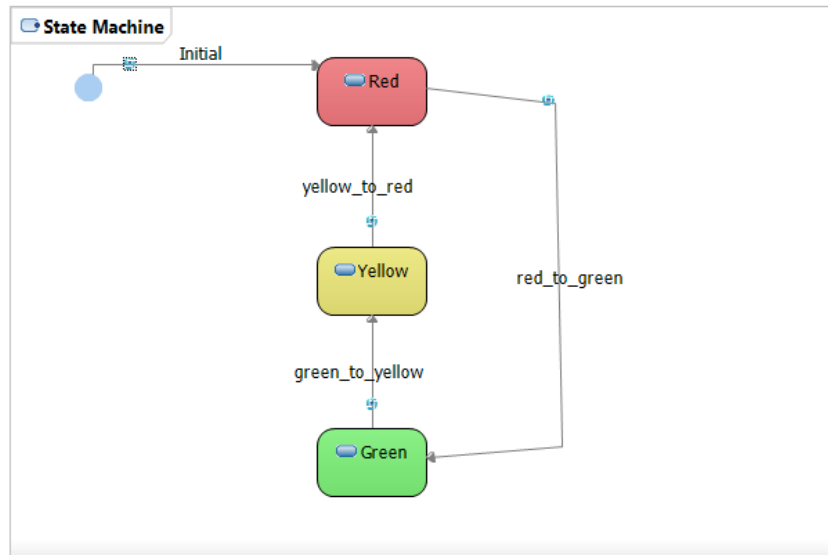
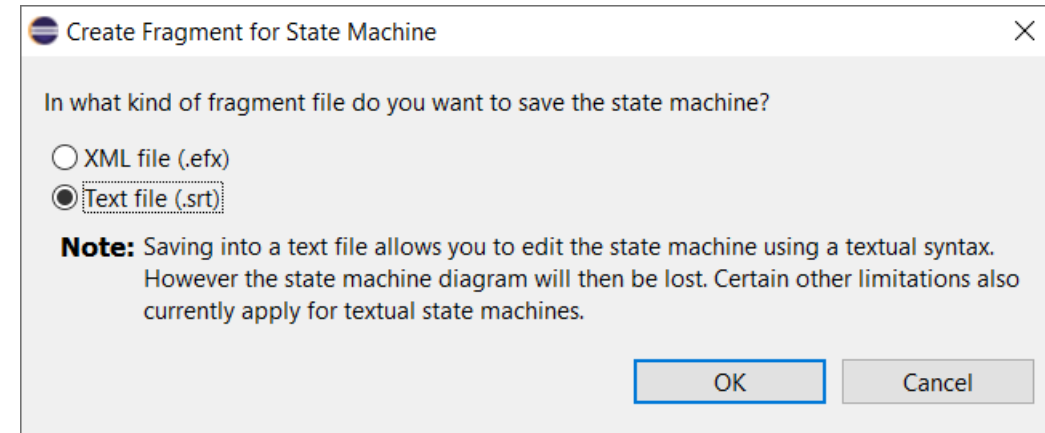
- ▶ The EGit integration in RTist has upgraded EGit from 5.8 to 5.12
 - This is the recommended and latest version for Eclipse 2021.06
- ▶ This upgrade provides several new features and bug fixes
 - For detailed information about the changes see
 - https://wiki.eclipse.org/EGit/New_and_Noteworthy/5.9
 - https://wiki.eclipse.org/EGit/New_and_Noteworthy/5.10
 - https://wiki.eclipse.org/EGit/New_and_Noteworthy/5.11
 - https://wiki.eclipse.org/EGit/New_and_Noteworthy/5.12

- ▶ RTist now supports to define capsule state machines using a textual language
 - The language includes all necessary state machine constructs, including a way to embed C++ code snippets
- ▶ This can sometimes be an attractive alternative to using graphical diagrams
 - Certain tasks are faster to perform using a textual language, for example copy/paste, refactoring etc.
 - It can be useful to have all C++ code snippets shown and edited in a single text editor (e.g. makes searching in those code snippets easier)
 - Possible to define textual templates for commonly used state machine constructs
 - Easier to merge changes in textually defined state machines

```
text-sm.srt x tl.srt TrafficLight.srt
1 statemachine 'State Machine' {
2   state State1, State2;
3   Initial: initial -> State1;
4   state Composite {
5     entry
6     `
7     std::cout << "Hello World!";
8     `;
9     exit
10    `
11    // Exited
12    `;
13    entrypoint ep1;
14    exitpoint ex1;
15  };
16  State1 -> State2 on timing.timeout when `return isAvailable()`
17  `std::cout << "Triggered!";`
18  ;
19 };
20
```

► Existing (graphical) state machines can be converted to a textual representation

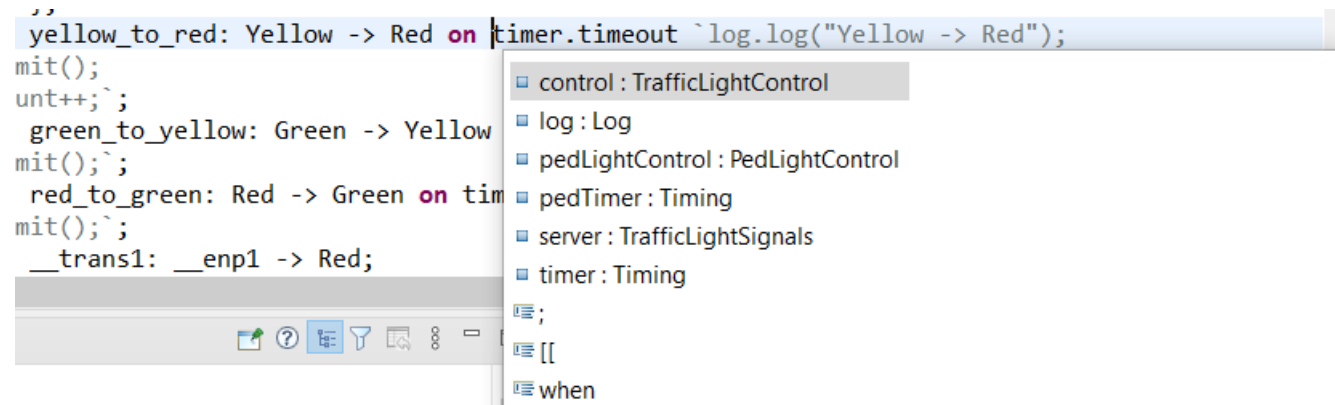
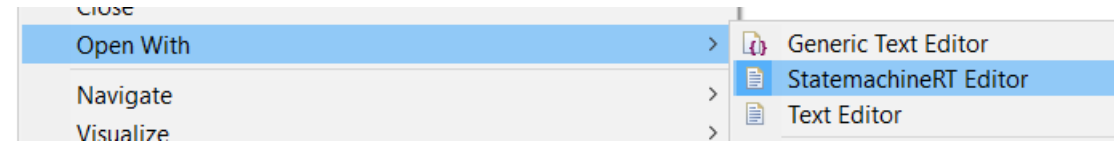
- Create a fragment for the state machine and store it in an .srt text file
- Note that existing state chart diagrams for the state machine will be lost (but can later be created from the textual representation for visualization purposes)



```
tl.srt x
1 statemachine 'State Machine' {
2   state Red;
3   state Green;
4   state Yellow;
5   Initial: initial -> Red `log.log("TrafficLight starts up");`;
6   red_to_green: Red -> Green on control.green `log.log("Red -> Green");`;
7   green_to_yellow: Green -> Yellow on control.yellow `log.log("Green -> Yellow");`;
8   yellow_to_red: Yellow -> Red on control.red `log.log("Yellow -> Red");`;
9 };
```

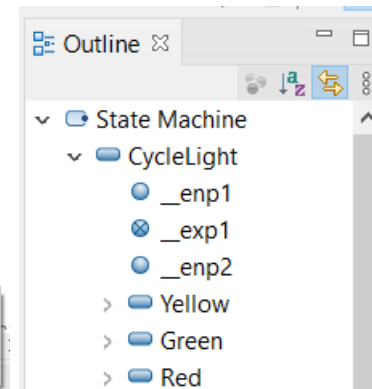
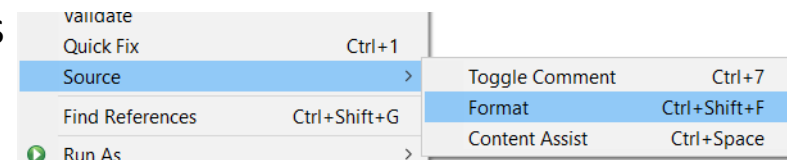
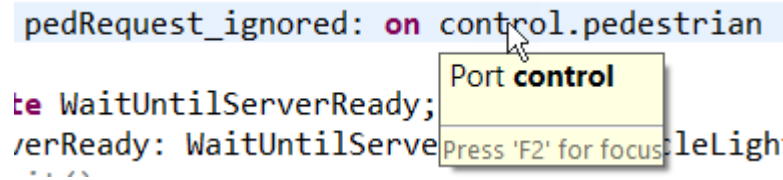
▶ Textual state machines can be edited in any text editor, but are best edited in the "StatemachineRT" text editor provided in RTist (associated in Eclipse with the .srt file extension)

- Ctrl + space content assist (provides both code templates for creating new elements, and automatic completion of names)
- Ctrl + click for navigating from a name reference to the corresponding definition (in the Project Explorer, or an in the same or a different .srt file)



```
pedRequest_ignored: on control.pedestrian`// Igr
```

- Tooltips when hovering over names
- Syntax coloring and folding
- Semantic validations assist creating correct state machines
- Outline view for overview and navigation
- Formatting and other useful commands are available in the context menu



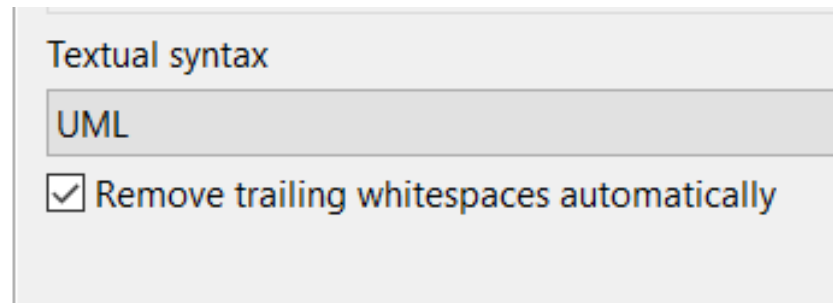
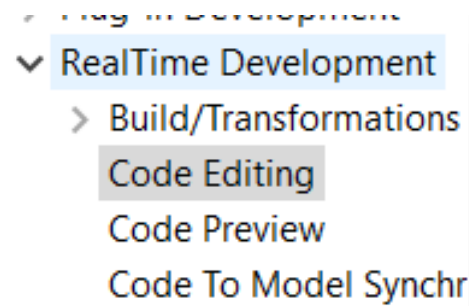
- ▶ The textual state machine is automatically updated if the underlying model changes (for example using the Project Explorer, Properties view or a state chart diagram)
- ▶ The model compiler supports textual state machines in the same way as graphical ones
 - Generated code is identical, and the only difference is some additional printouts in the build log

```
16:29:06 : INFO : Loading root models
16:29:09 : INFO : EXPERIMENTAL: Textual statemachine: TrafficLight.srt <-- TrafficLight
16:29:09 : INFO : EXPERIMENTAL: Loading SRT file:/D:/eclipse-workspace/rtist-11-2-wksp/TrafficLightsDemo/TrafficLight.srt
```

- ▶ Note that the support for textual state machines is currently an experimental feature and certain limitations exist
 - Code-to-model synchronization is not available
 - RTist search commands do not index .srt files (but regular file-based search in Eclipse works, as well as search within an .srt file)
 - Not integrated with the RTist Compare/Merge editor (but regular Eclipse compare/merge works)
 - There are known issues with using both a graphical diagram and the text editor for editing a textual state machine
 - Support for textual state machines affects the keybinding to open the Code Editor (Ctrl+Shift+F3)
 - Passive class state machines are not supported. Only capsule state machines can be textually defined.

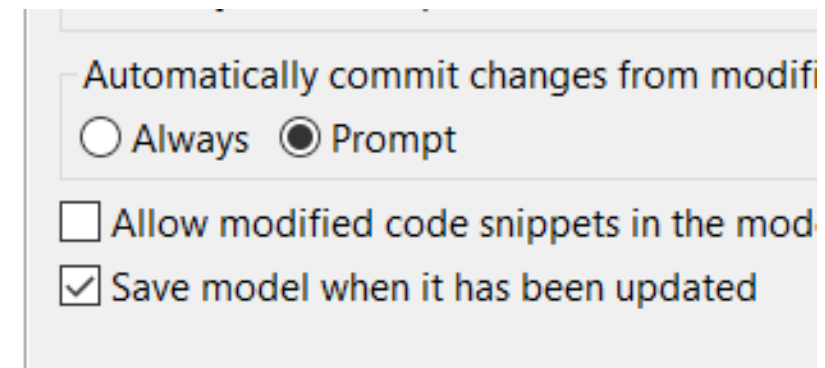
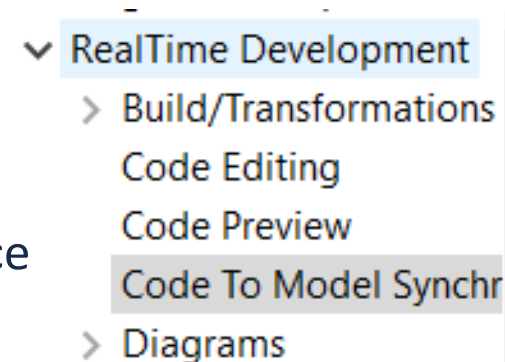
Model Editing Improvements

- ▶ Trailing whitespaces can be automatically removed on save in Code editor and on committing model change in Code view, if corresponding setting is enabled



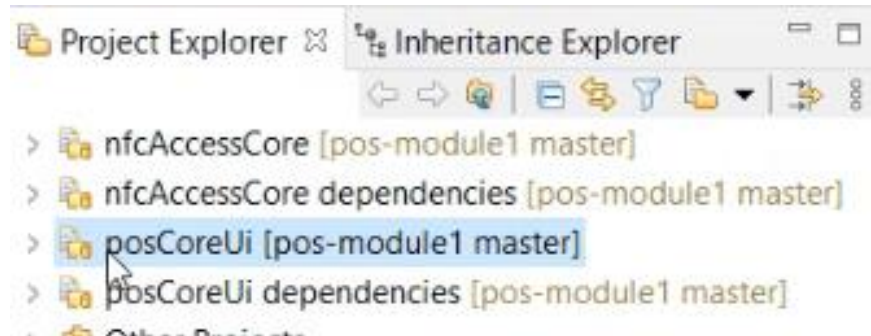
- ▶ Standard copy command now copies names of selected elements and diagrams, that later can be pasted into other editors and applications.

- ▶ Initial state of "Save model when it has been updated" checkbox in Code to Model Synch dialog can be controlled with a new preference

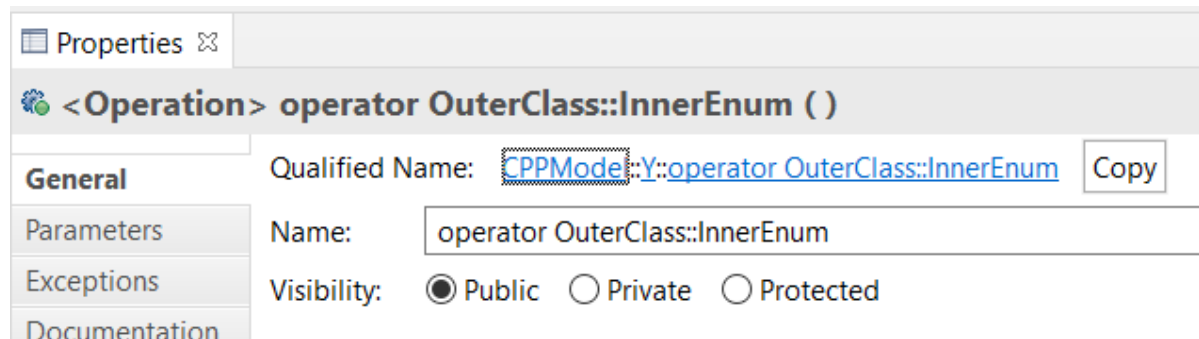


Other Improvements

- ▶ External Projects importer will now automatically create working 2 sets: one for projects being imported and the other for their dependencies derived from map or mapping files



- ▶ Operations with :: in the name are supported



HCL

*Relationship*TM
BEYOND THE CONTRACT

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



WATCH THE FILM