


What's New in HCL RTist 11.2

updated for release 2022.41

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.v20221014_0755
Release: 2022.41

(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>

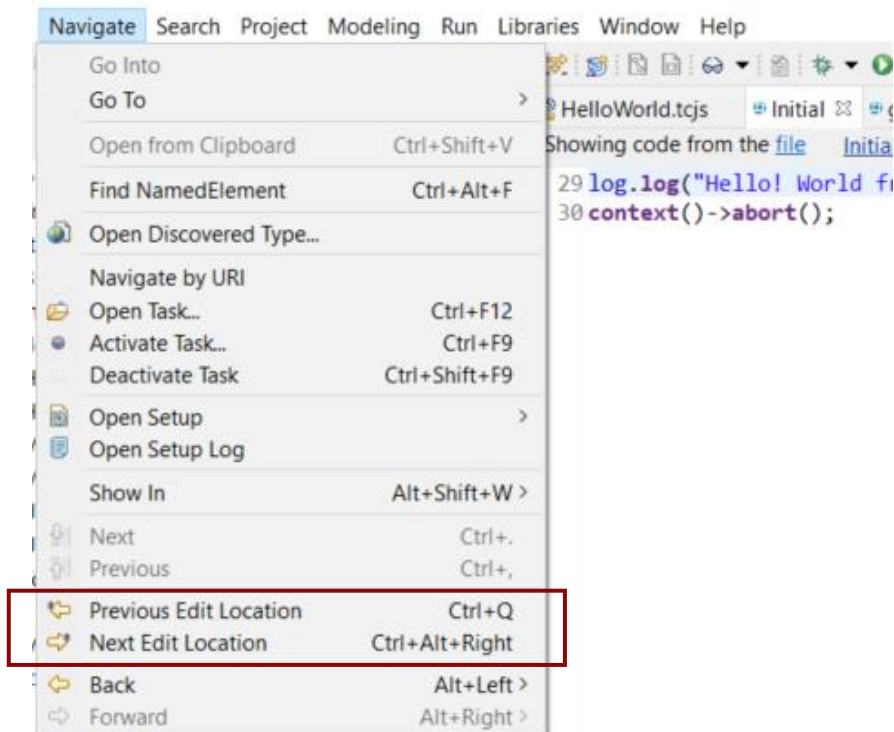


Eclipse 4.20 (2021.06)

- ▶ 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...

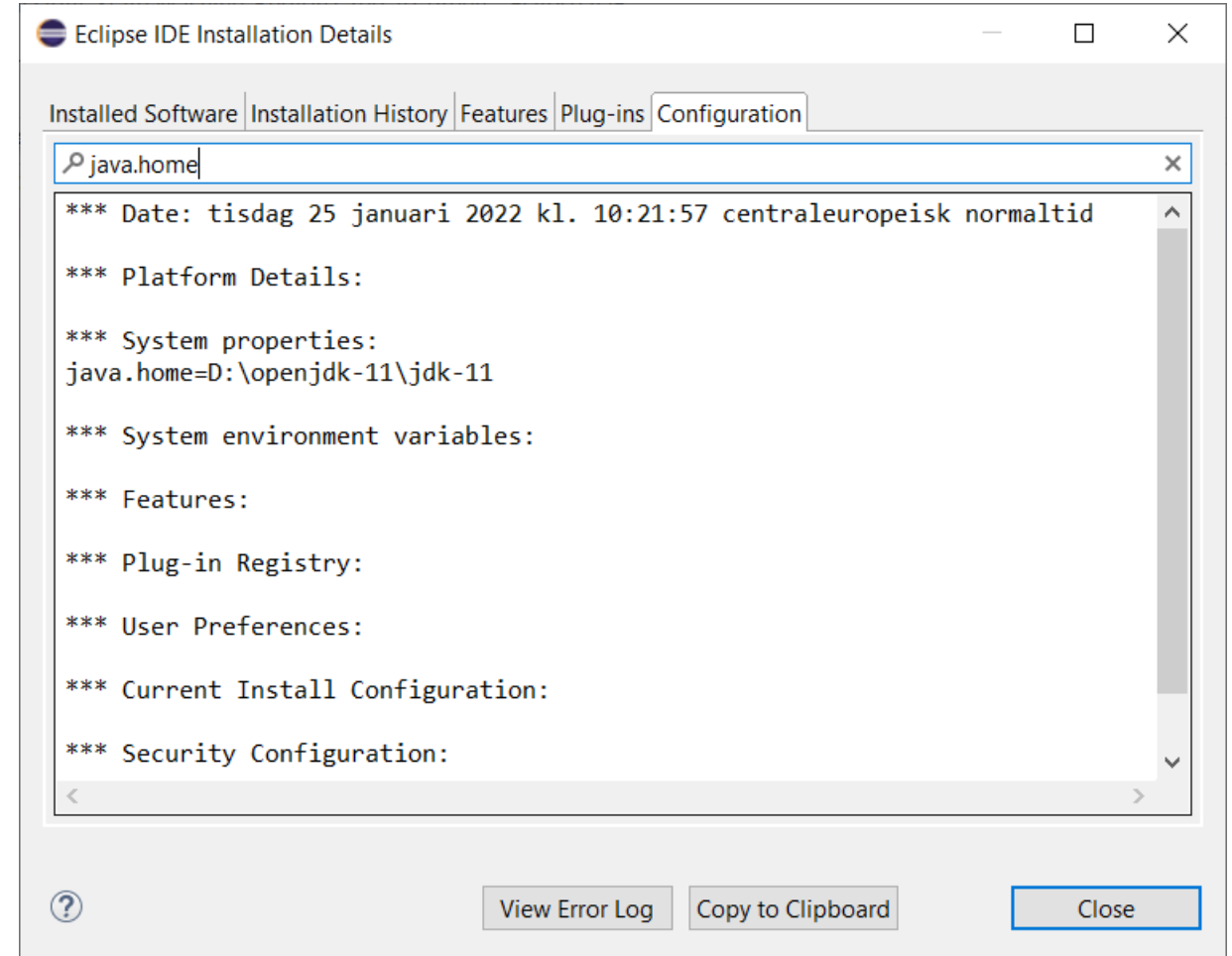
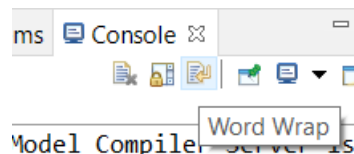
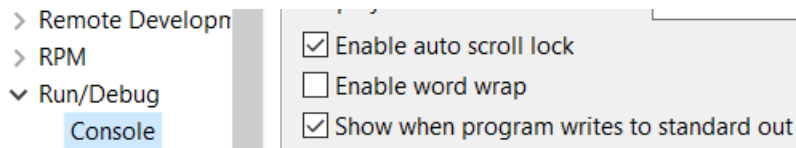
Eclipse 4.20 (2021.06)

- ▶ 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)



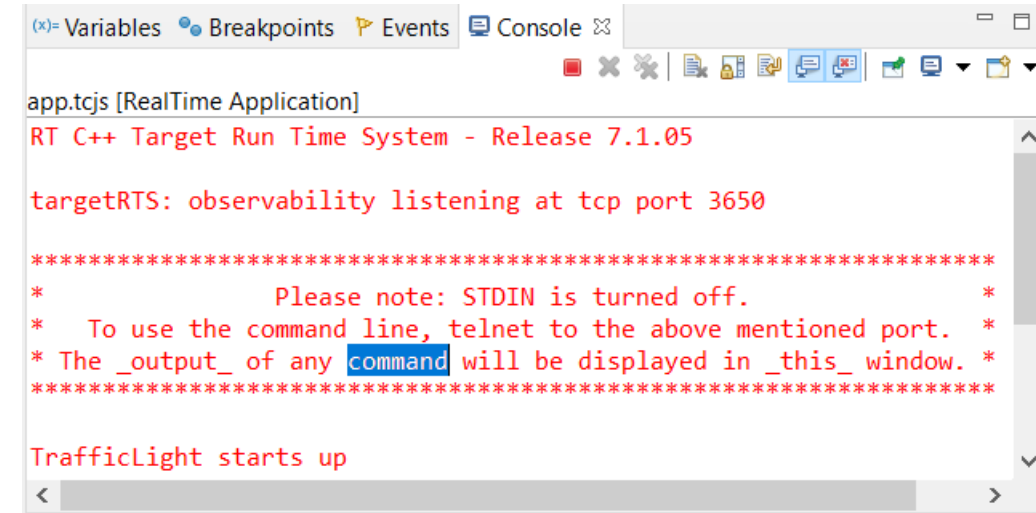
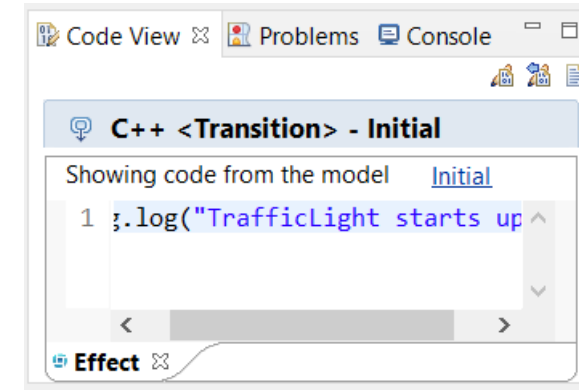
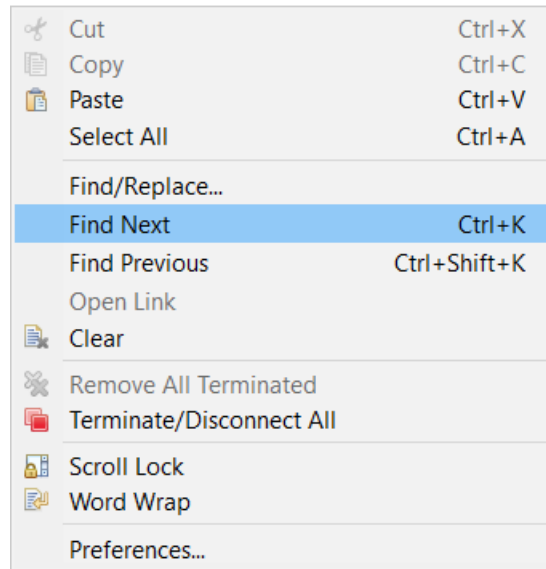
Eclipse 4.20 (2021.06)

- ▶ 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



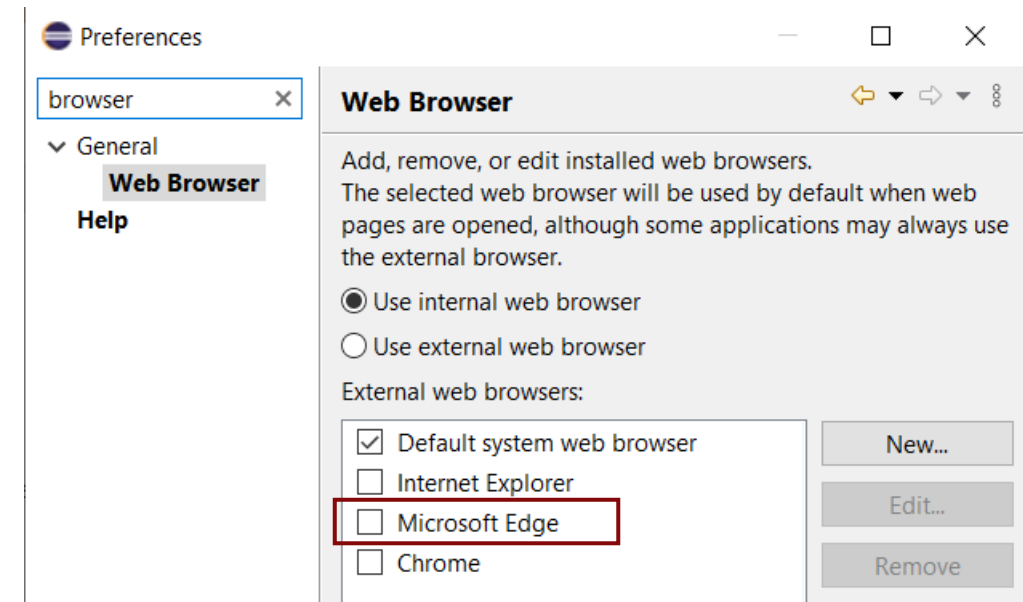
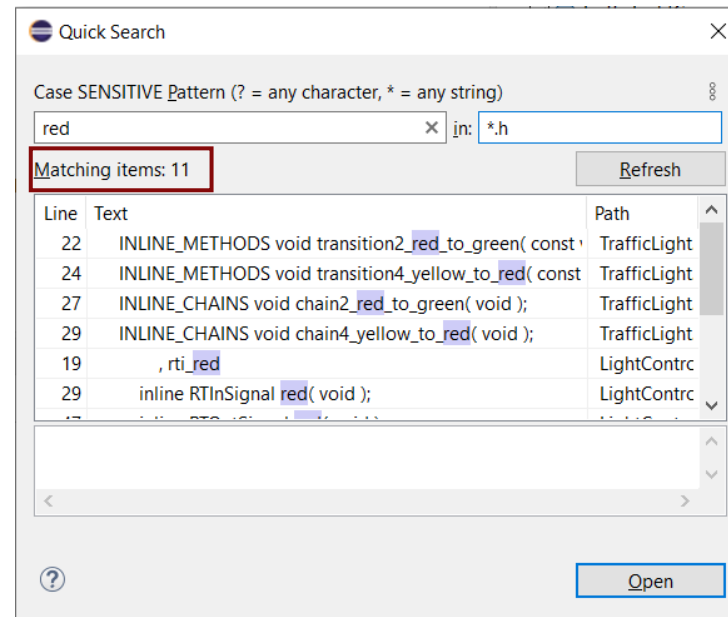
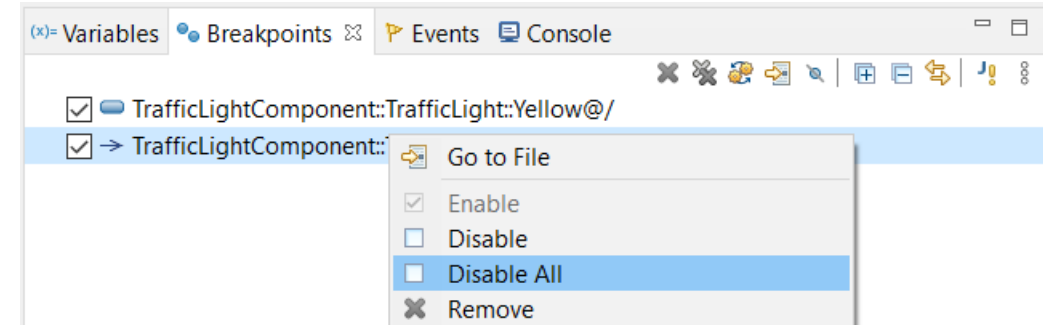
Eclipse 4.20 (2021.06)

- ▶ 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



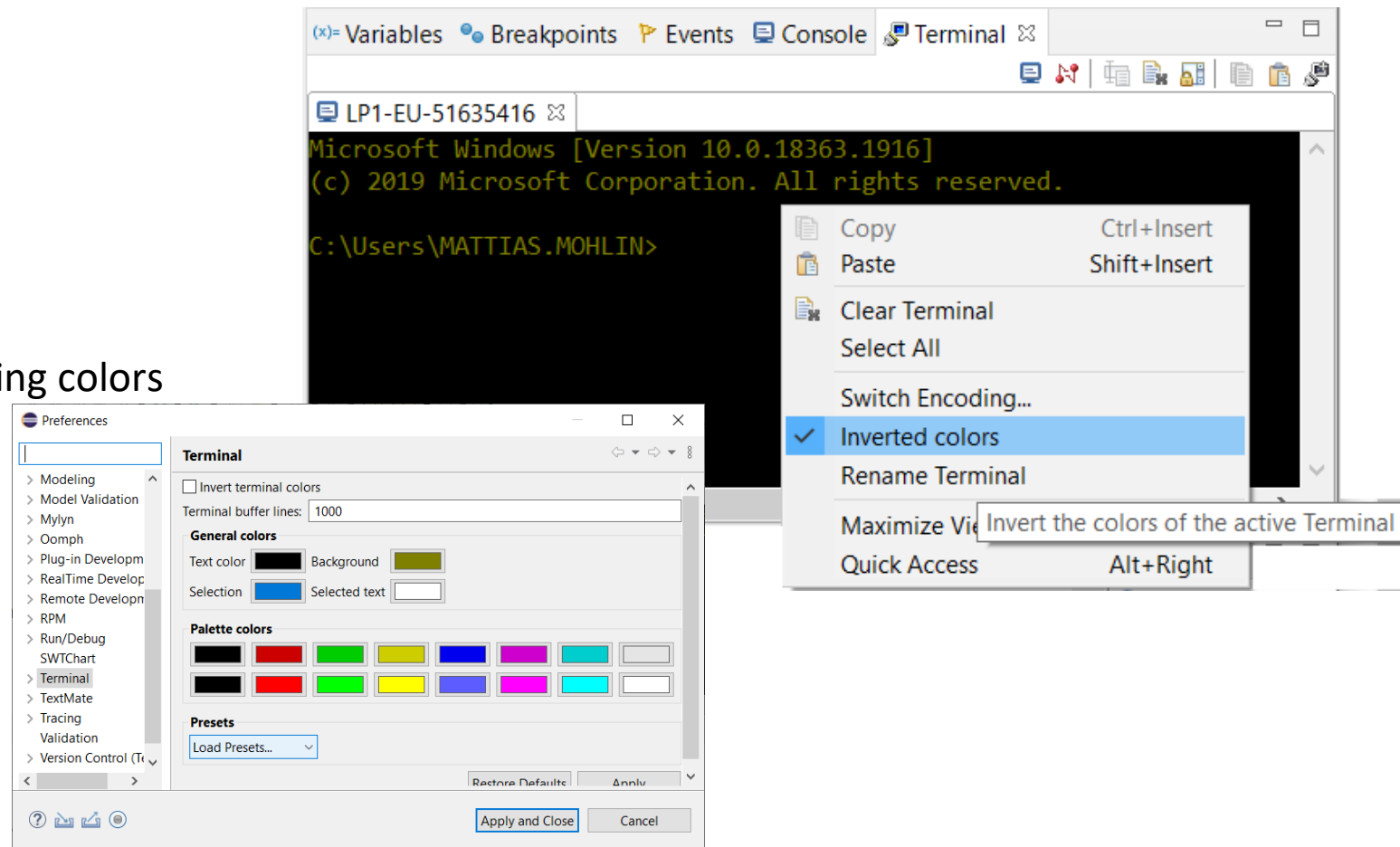
CDT 10.3 (included as part of Eclipse 2021.06)

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

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

▶ More configurable Terminal view

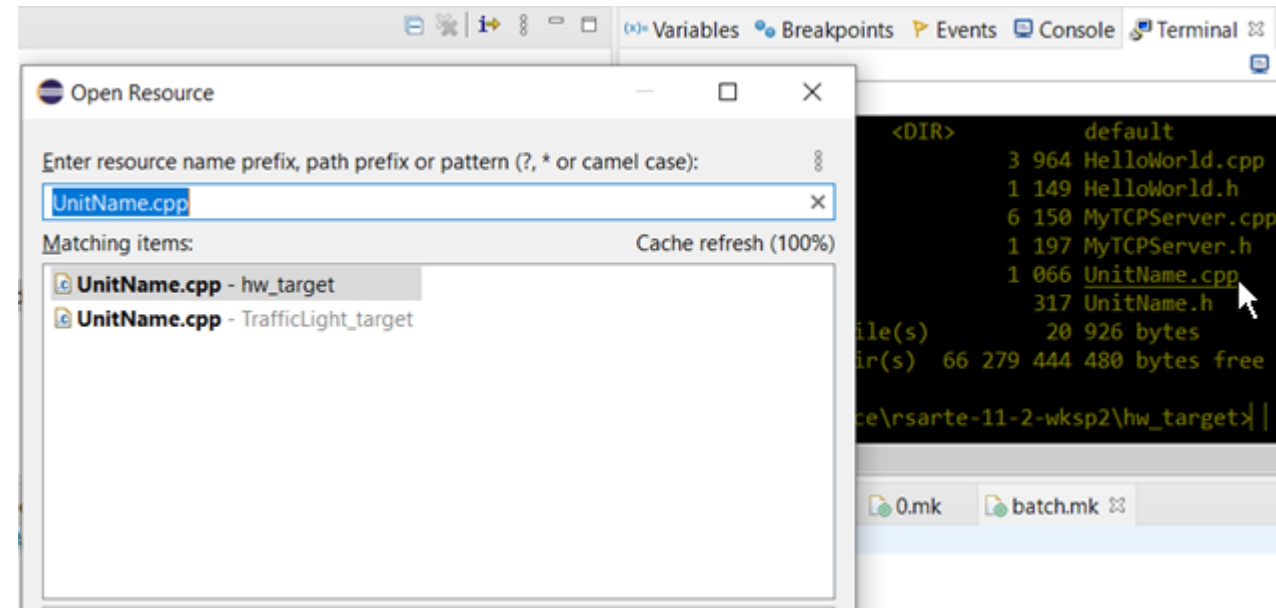
- New context menu command for inverting colors
- New preferences for configuring the colors used by the Terminal view
- Changing the colors helps for example when connecting to certain remote systems that make assumptions about what colors are used
- New context menu command for renaming the terminal (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

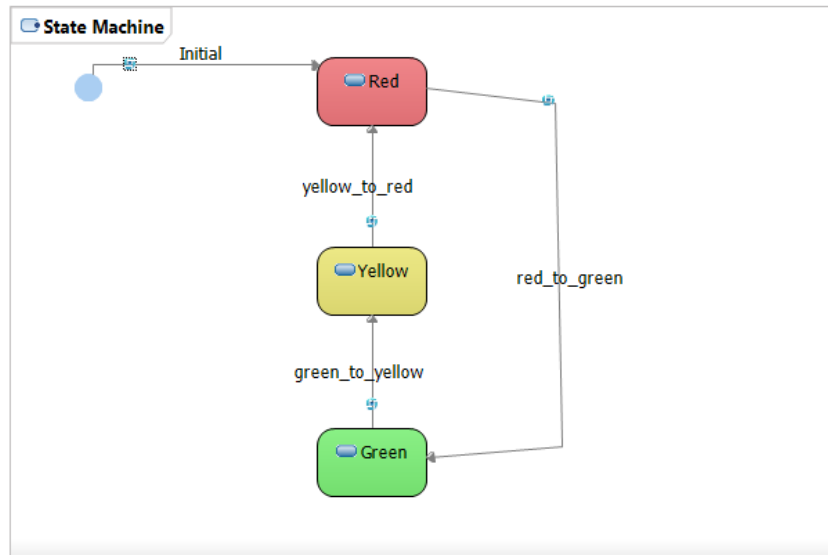
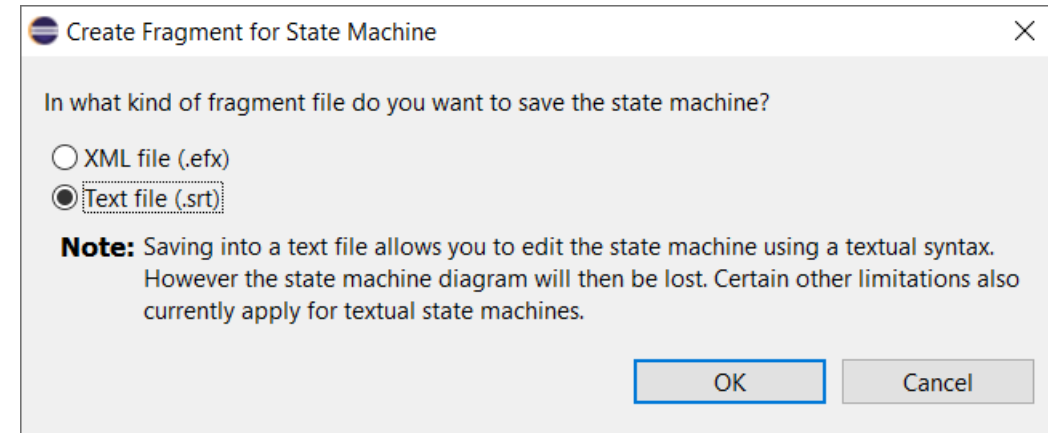
Textual State Machines (1/4)

- ▶ 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
```

Textual State Machines (2/4)

- ▶ 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 };
```

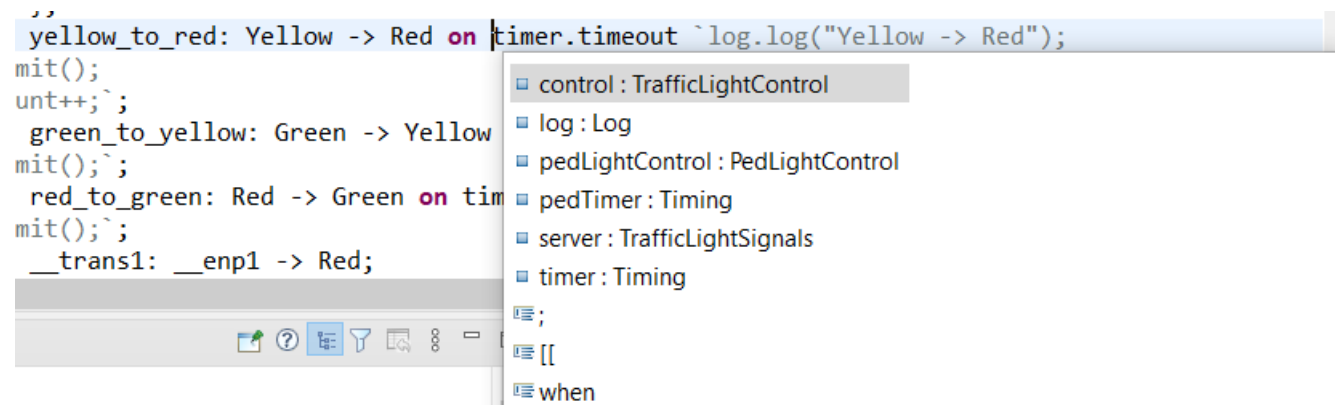
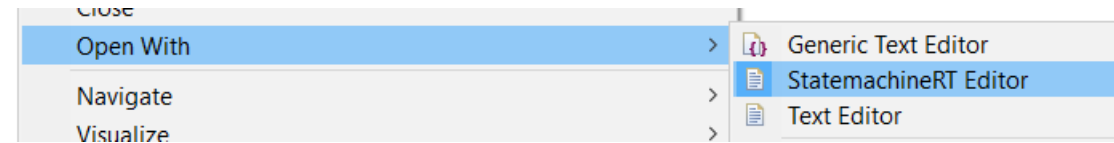
Textual State Machines (3/4)

- ▶ 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)

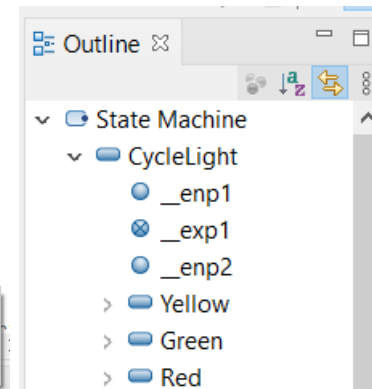
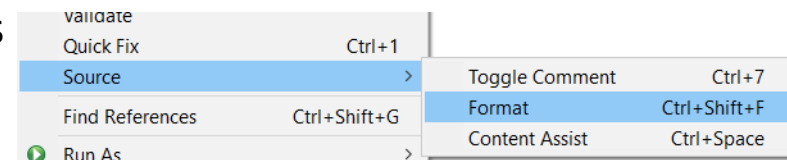
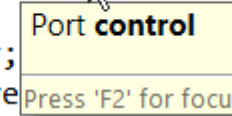
- Ctrl + space content assist (provides both code templates for creating new elements, and automatic completion of names)
- Ctrl + click for navigating from a name reference to the corresponding definition (in the Project Explorer, or an in the same or a different .srt file)

```
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



```
pedRequest_ignored: on control.pedestrian
```



Textual State Machines (4/4)

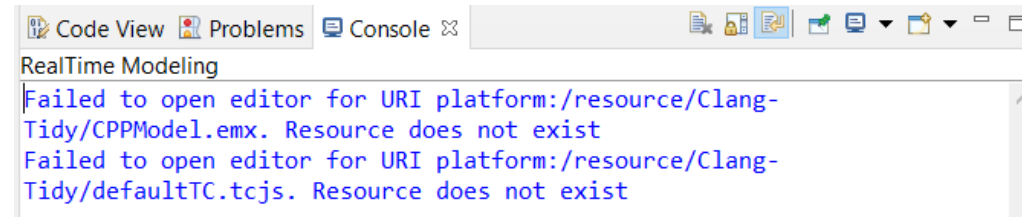
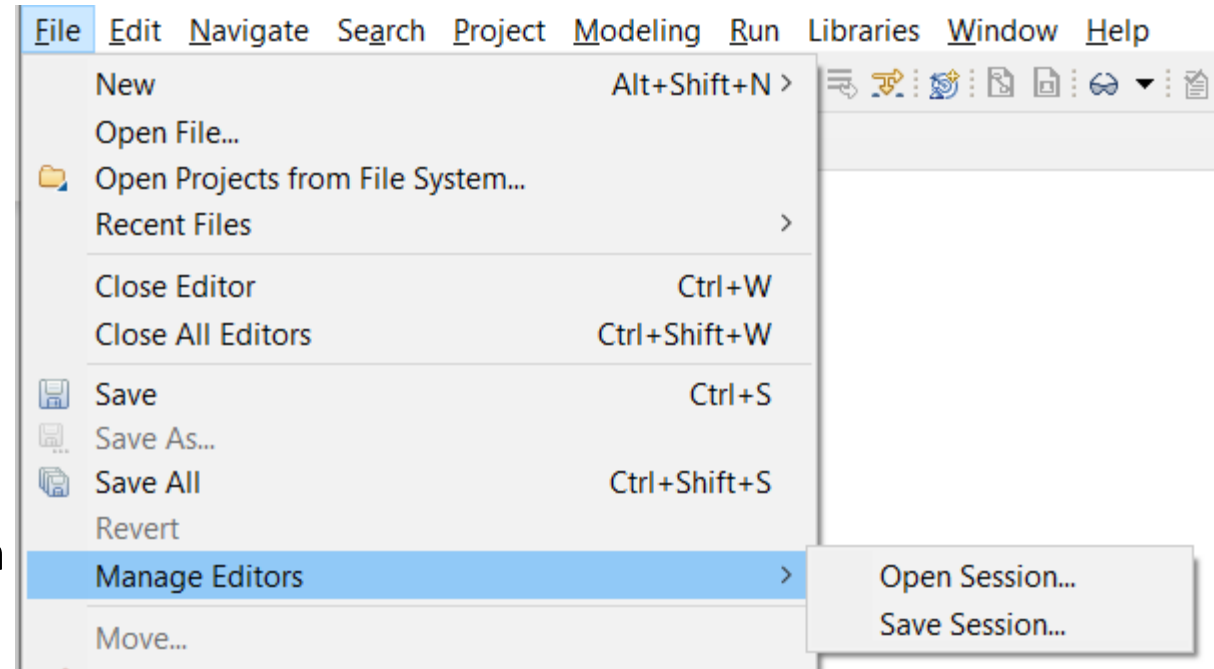
- ▶ The textual state machine is automatically updated if the underlying model changes (for example using the Project Explorer, Properties view or a state chart diagram)
- ▶ The model compiler supports textual state machines in the same way as graphical ones
 - Generated code is identical, and the only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

Manage Editors

- ▶ New commands are available for more easily restoring the set of open editors
 - **Save Session**
Saves information about open editors to a .session file
 - **Open Session**
Open editors from information in a .session file
- ▶ Useful for example after restarting RTist, when switching between different working tasks, or when there is a need to temporarily close all open editors and later open them again (e.g. when performing commands that change several model files on disk, to avoid time-consuming refreshing)
- ▶ Messages printed by the commands are shown in the console
 - For example if the resource of an opened editor is not present in the workspace



Drag/drop Support from More Views

- ▶ Now it's possible to drag/drop elements from other views than the Project Explorer to diagrams

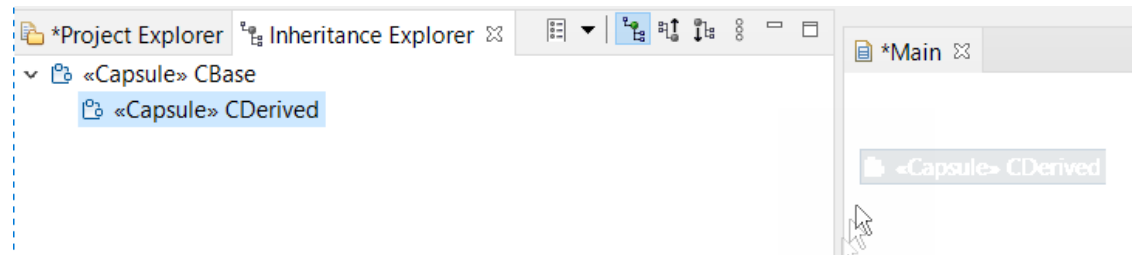
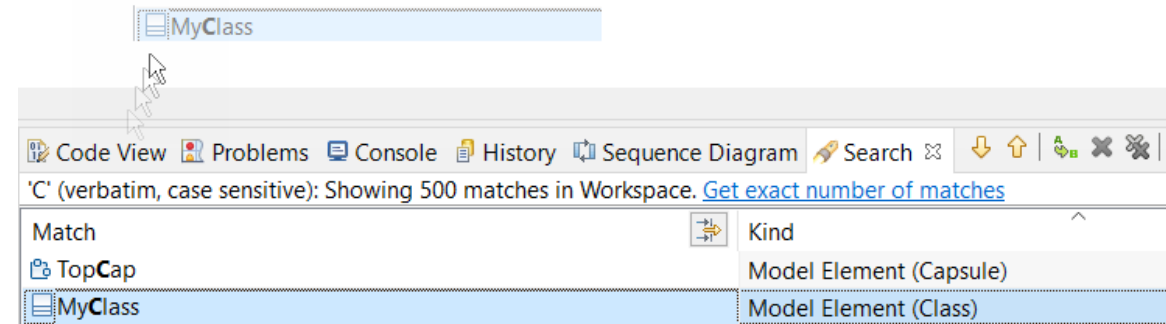
- Search view
- Inheritance Explorer

- ▶ Previously this was a two-step process: first navigate from the Search view / Inheritance Explorer to the Project Explorer and then perform the drag/drop

- Now the navigation step is avoided

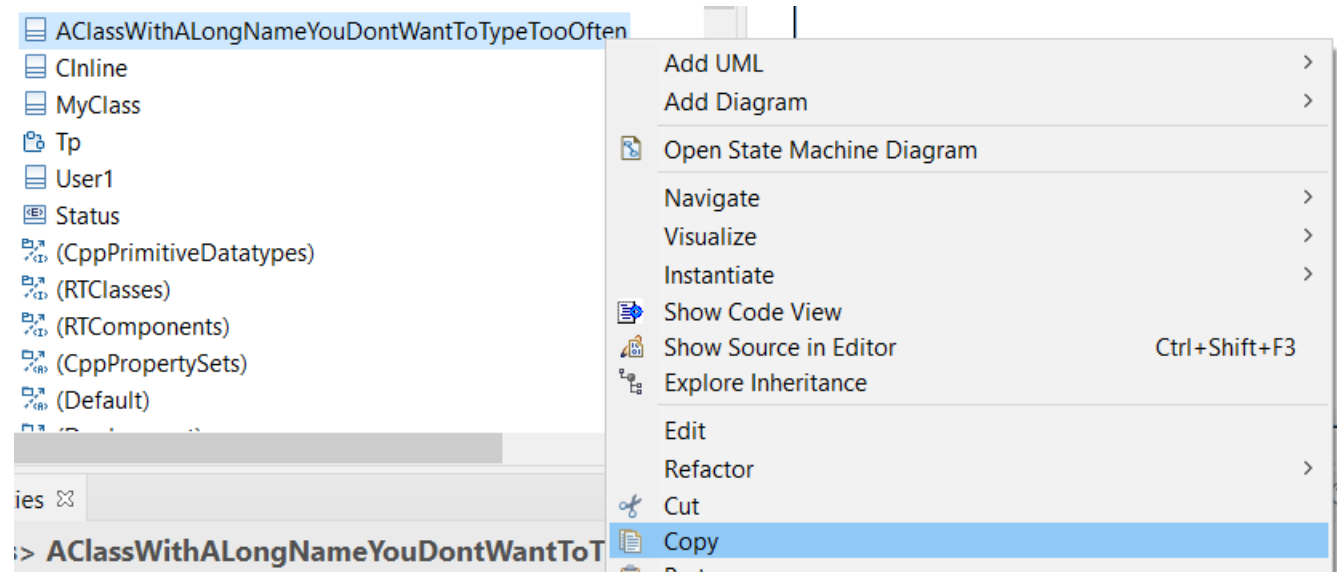
- ▶ Known limitation: Dragging a capsule to a composite structure diagram in order to create a part for it doesn't work

- In this particular case, it's still necessary to drag from the Project Explorer



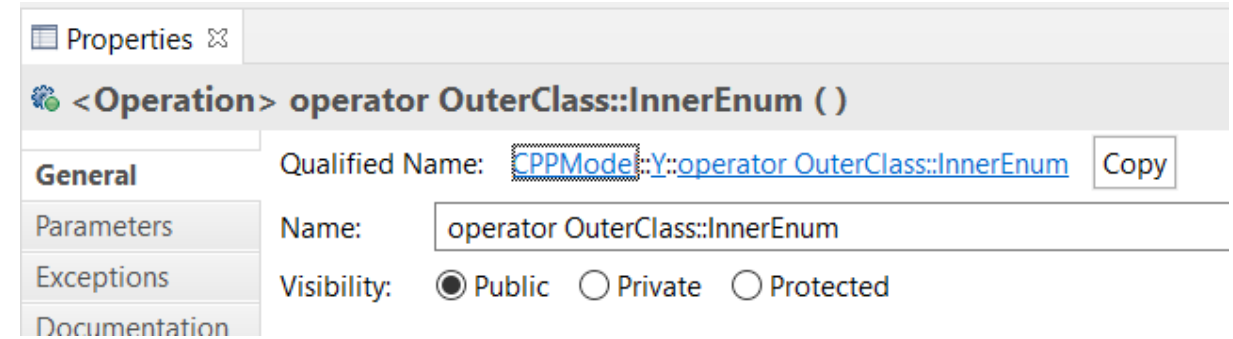
Copy/Paste of Element Names

- ▶ All elements with a name (model elements, diagrams) can now have their names copied from the Project Explorer
- ▶ Select the element in the Project Explorer and perform the Copy command (Ctrl + C)
- ▶ The name is placed in the clipboard and can be pasted in any text editor (e.g. the Code view or Code editor)
- ▶ Avoids the need to type long names in code snippets
- ▶ Note that this was already before supported for Eclipse projects, and several other elements shown in the Project Explorer



Better Handling of C++ Names and Types

- ▶ It's now possible to use names that contain ::
 - For example necessary when defining a conversion operator function for a nested type
- ▶ Parsing of C++ types were improved
 - Commas are now supported – needed for instantiations of templates with multiple template parameters
 - Nested template instantiations are now also supported



Support for inline variables

- ▶ A static member variable or global variable that is defined in the header file needs to be declared as **inline**; otherwise the header file cannot be included from multiple compilation units without getting a linker error
- ▶ A new checkbox was added for declaring an attribute as inline
- ▶ Note that inline variables requires C++ 17 or later
 - The model compiler will detect if you attempt to use this feature with a too old language standard

The image shows two screenshots from an IDE. The top screenshot shows the properties of an attribute named 'x'. The 'General' tab is selected, and the 'Modifiers' section has the 'Inline' checkbox checked. The 'Type' is set to 'int' and the 'Default Value' is '5'. The bottom screenshot shows the properties of a class named 'CInline'. The 'Attributes' section shows the attribute 'x : int { 5 }' with the 'Inline' checkbox checked.

10:43:17 : WARNING : CPPModel::CInline::x : Inline variables require C++ 17 or later

Attribute Initialization in Copy Constructor

- ▶ The generated copy constructor now takes the property **Initializer Kind** of an attribute into account to decide how to initialize the attribute

- ▶ Brace initialization

```
MyClass  
  m_status : Status { Status::Success }
```

Initializer Kind:

```
MyClass::MyClass( const MyClass & rtg_arg )  
  : m_status{ rtg_arg.m_status }  
{  
}
```

- ▶ Equal / assignment initialization

```
MyClass  
  m_status : Status = Status::Success
```

Initializer Kind:

Initializer Kind:

```
MyClass::MyClass( const MyClass & rtg_arg )  
{  
  m_status = rtg_arg.m_status;  
}
```

- ▶ Constructor initialization

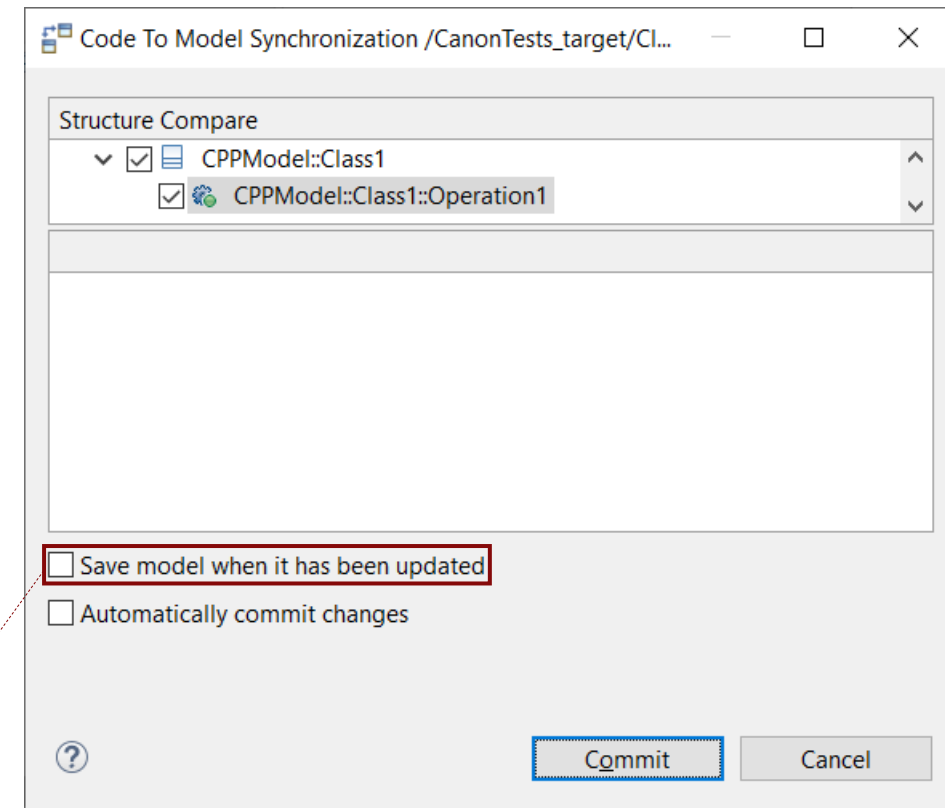
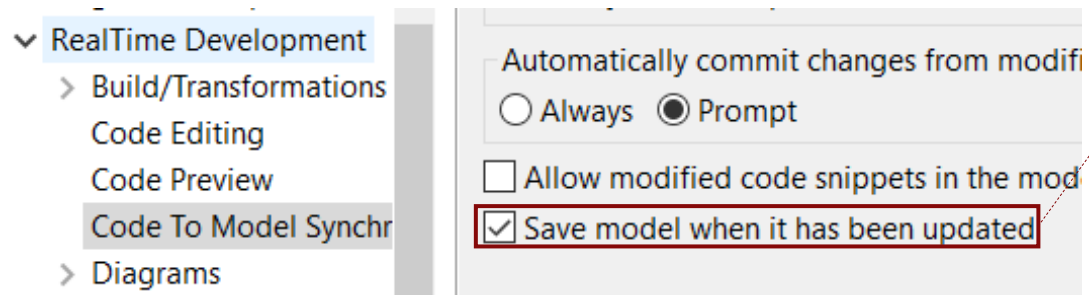
```
MyClass  
  m_status : Status = Status::Success
```

Initializer Kind:

```
MyClass::MyClass( const MyClass & rtg_arg )  
  : m_status( rtg_arg.m_status )  
{  
}
```

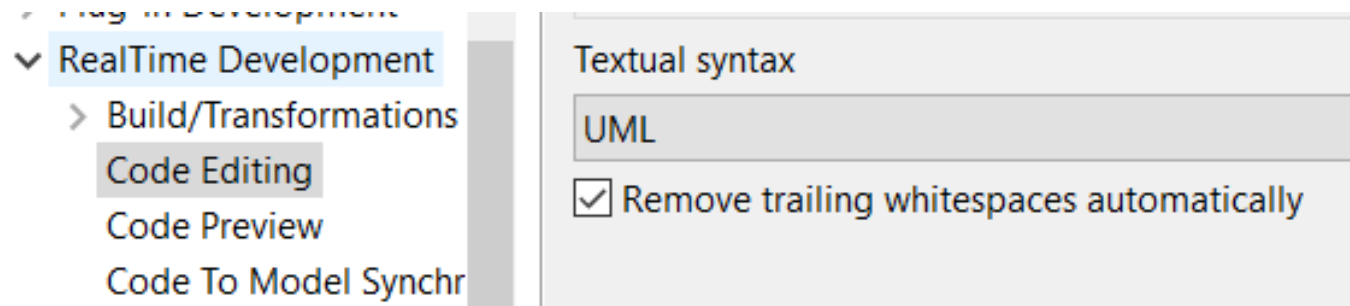
Code to Model Synchronization

- ▶ A new preference can be set for automatically saving the model after running code-to-model synchronization
 - **RealTime Development – Code To Model Synchronization – Save model when it has been updated**
- ▶ Previously this had to be set in the dialog, and was not remembered after restarting RTist
- ▶ The dialog checkbox still is present and can now be used as a shortcut for changing the preference



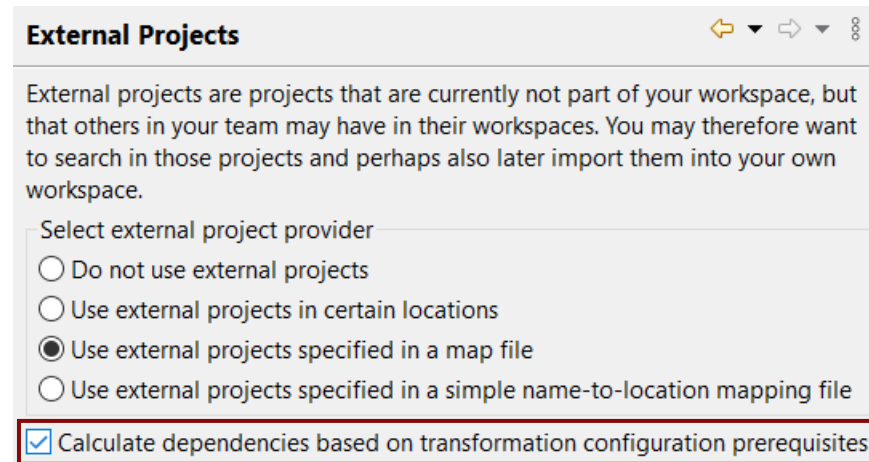
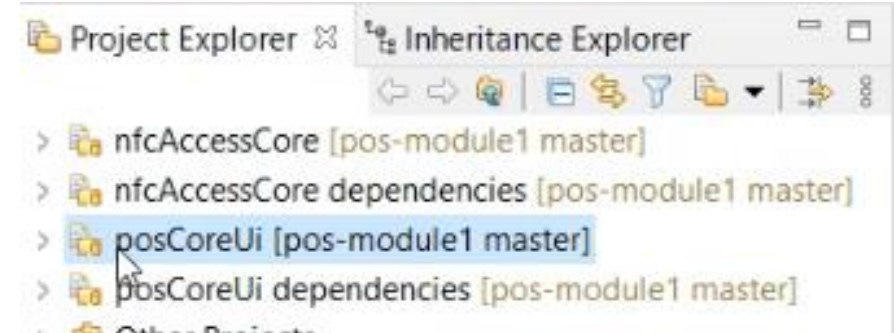
Automatic Removal of Trailing Whitespace from Code

- ▶ Eclipse CDT provides this feature for C++ files
 - Preference **C/C++ - Editor – Save Actions – Remove trailing whitespace**
- ▶ A similar preference was added to support the same for code snippets
 - Preference **RealTime Development – Code Editing – Remove trailing whitespaces automatically**
 - For the Code editor, white spaces are removed when you save it.
For the Code view, white spaces are removed when the changed code snippet is committed to the model.



External Project Importer

- ▶ Now two working sets will be automatically created; one for the projects being imported and another for their dependent projects
- ▶ A new preference was added for automatically calculating project dependencies by analyzing TC prerequisites
 - Avoids the need to manually maintain the information about dependent projects in the map file
 - Makes it possible to search in prerequisites of external projects



Transformation Configurations

- ▶ The "Workspace output path" property now allows for more flexibility
 - An absolute path can be used
 - Subfolders are supported
 - If the variable `$(TCONFIG_NAME)` is used for this property in an inherited TC, it's now correctly resolved with the name of the local TC
- ▶ The standard Eclipse variable `$(workspace_loc)` is now supported in several TC properties where paths can be specified
 - Expands to the location of the workspace
 - Will be generated into the makefile and can hence be used in all properties that are resolved when invoking 'make'

Target

Automatically create and update target project

Workspace output path:

Use default location

Location:

Visual Studio 2022

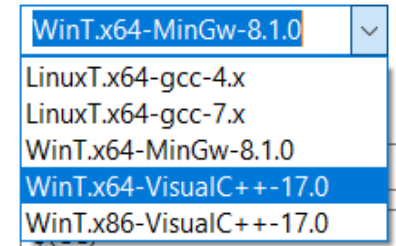
- ▶ The Visual Studio 2022 C++ compiler (ver 17.0) can now be directly used with RTist
 - A target configuration for this compiler is available
 - Prebuilt libraries for this compiler are included (TargetRTS, Connexis, LibTCPServer)
 - Replaces the old Visual Studio 2017 (ver 15.0) target configurations and libraries
 - Both 64 and 32 bit versions are available

TargetRTS configuration:

Make type:

Compile arguments:

Compile command:



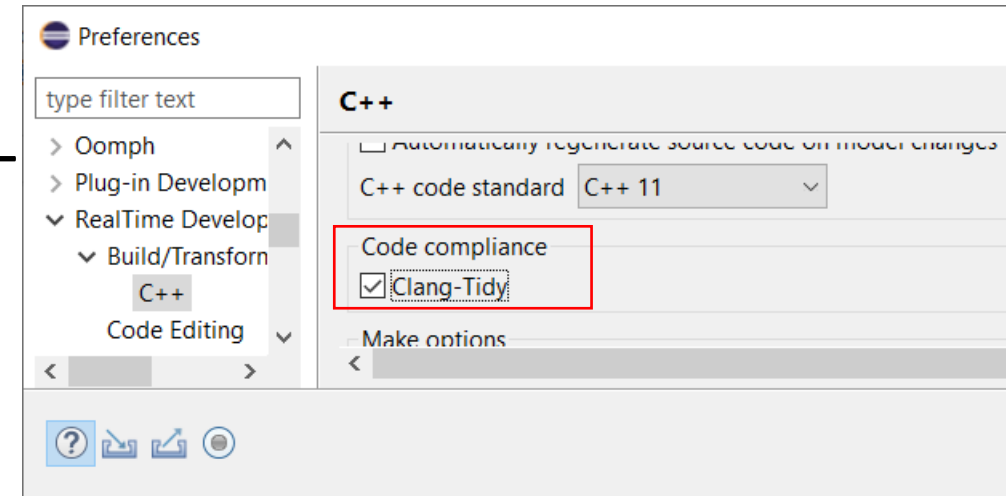
Code Compliance

- ▶ An additional Clang-Tidy rule is now supported when the preference **RealTime Development – Build/Transformations – C++ - Clang-Tidy** is set:

- **google-explicit-constructor**

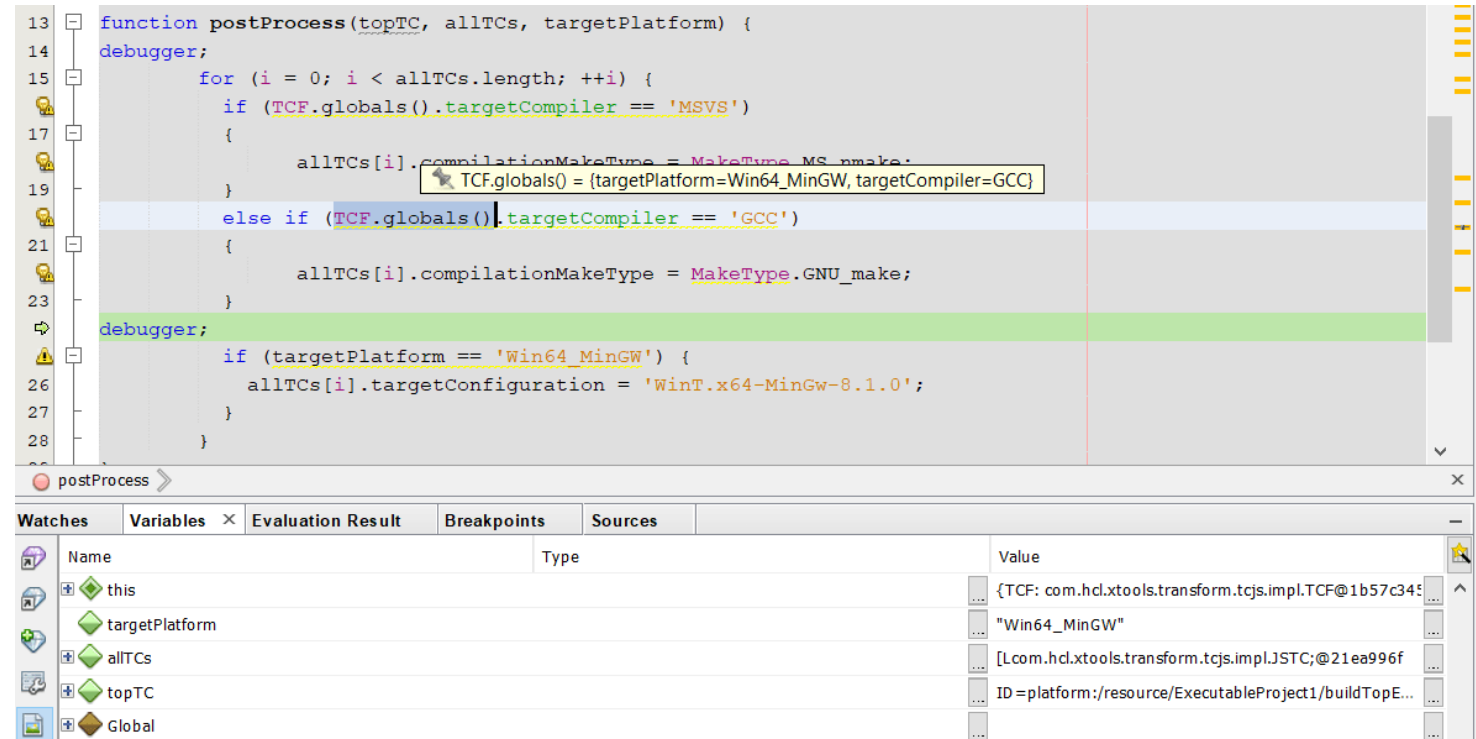
Suppress warnings for type descriptor constructors that are used for implicit conversions from a type with a type descriptor to the RTTypedValue struct for that type.

```
inline RTTypedValue_MyClass( const MyClass & rtg_value /* NOLINT(google-explicit-constructor) */ )
```



Debugging Build Variant Scripts

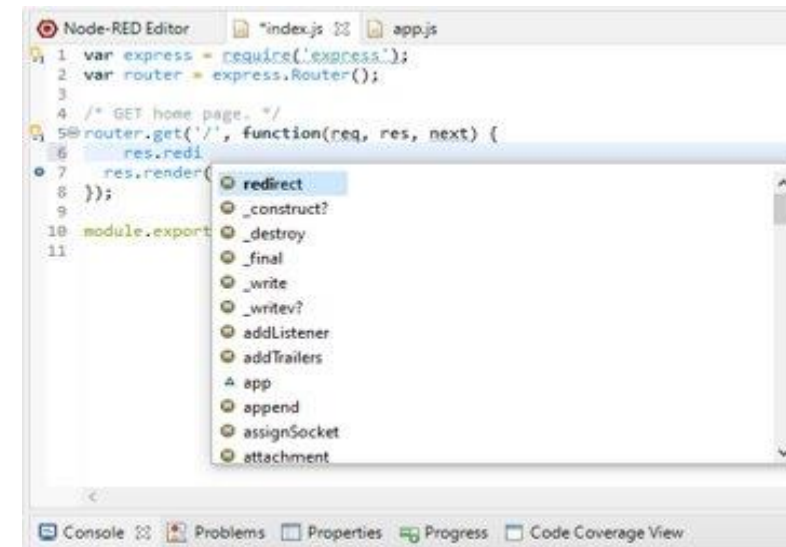
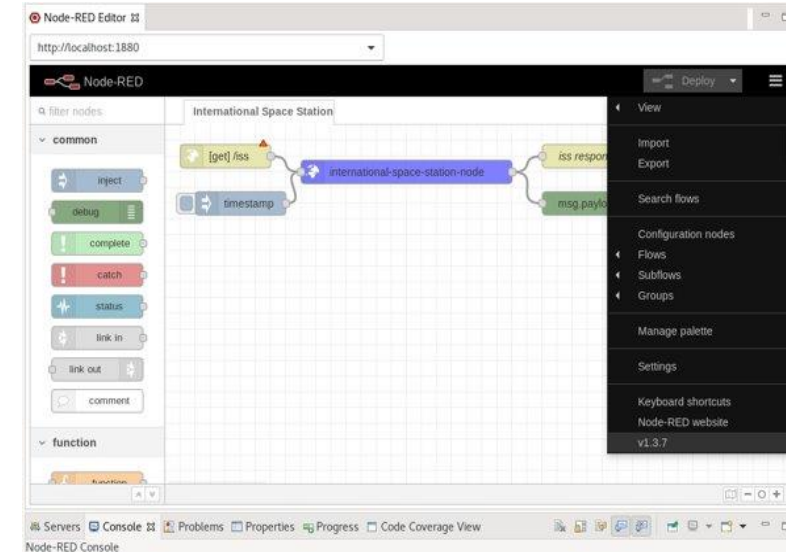
- ▶ Build Variant Scripts can be debugged using a debugger that supports Nashorn
 - The NetBeans IDE has this support
- ▶ A new article in the documentation was written with step-by-step instructions for how to debug build variant scripts using NetBeans:
 - Debugging works both for scripts that run in the context of the Eclipse UI, and scripts that run in the context of the model compiler



```
13 function postProcess(topTC, allTCs, targetPlatform) {
14     debugger;
15     for (i = 0; i < allTCs.length; ++i) {
16         if (TCF.globals().targetCompiler == 'MSVS')
17         {
18             allTCs[i].compilationMakeType = MakeType.MS_make;
19         }
20         else if (TCF.globals().targetCompiler == 'GCC')
21         {
22             allTCs[i].compilationMakeType = MakeType.GNU_make;
23         }
24     }
25     debugger;
26     if (targetPlatform == 'Win64_MinGW') {
27         allTCs[i].targetConfiguration = 'WinT.x64-MinGw-8.1.0';
28     }
29 }
```

Name	Type	Value
this		{TCF: com.hcl.xtools.transform.tcjs.impl.TCF@1b57c345}
targetPlatform		"Win64_MinGW"
allTCs		[Lcom.hcl.xtools.transform.tcjs.impl.JSTC;@21ea996f
topTC		ID=platform:/resource/ExecutableProject1/buildTopE...
Global		

- ▶ NodePlus 2.0 now supports Eclipse 2021.06 and can be used with RTist 11.2
- ▶ It's now delivered as a separate update site to reduce the size of the RTist update site
- ▶ NodePlus now uses the Wild Web Developer (WWD) which provides a rich development environment for the typical languages used for web development (JavaScript, CSS, HTML, etc)
- ▶ The Node-RED server was updated
- ▶ NodePlus is still only available for RTist users, but work is ongoing for supporting also IBM RSARTE in the near future



HCL

*Relationship*TM
BEYOND THE CONTRACT

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

 WATCH THE FILM