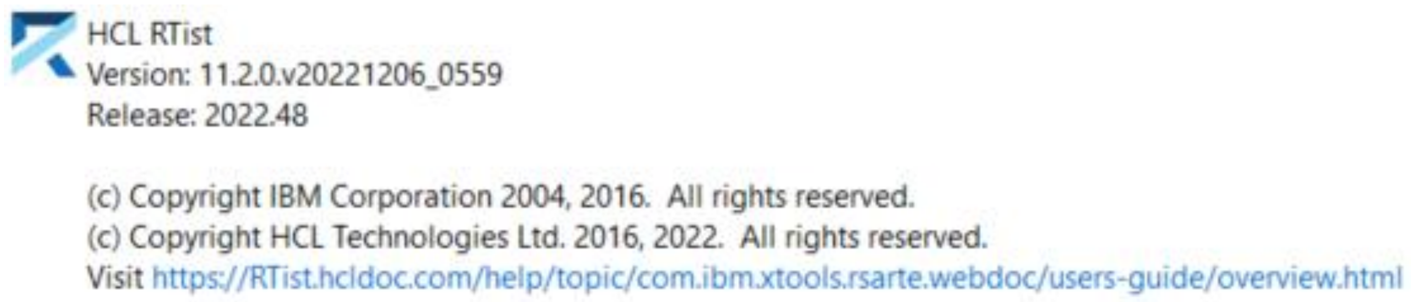
What’s New in HCL RTist 11.2

updated for release 2022.48

HCL SOFTWARE

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 RTist only


HCL RTist
Version: 11.2.0.v20221206_0559
Release: 2022.48

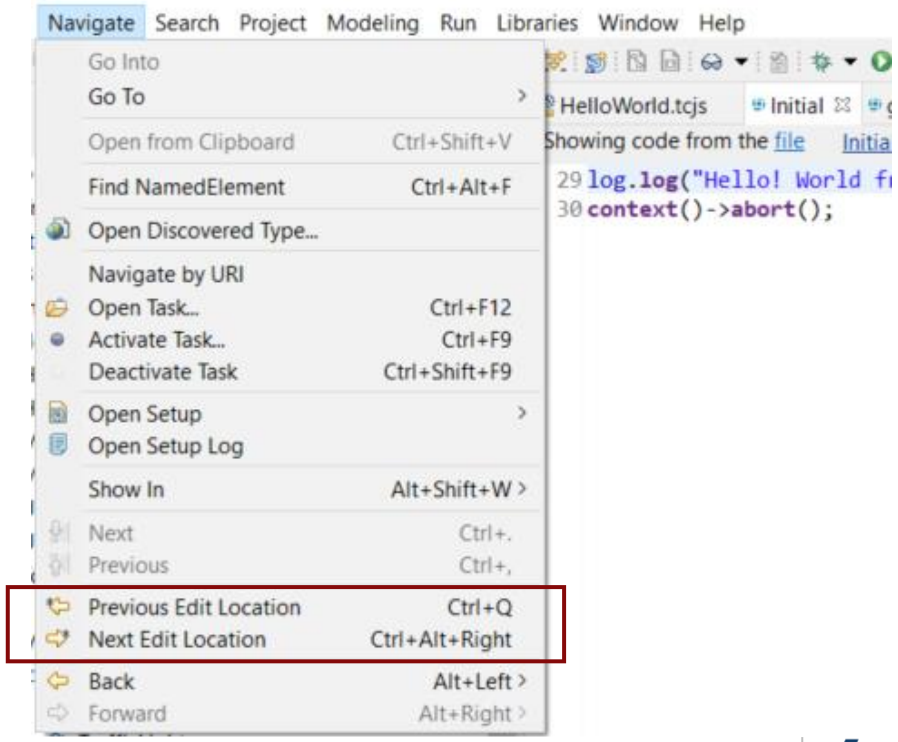
(c) Copyright IBM Corporation 2004, 2016. All rights reserved.
(c) Copyright HCL Technologies Ltd. 2016, 2022. All rights reserved.
Visit https://RTist.hcldoc.com/help/topic/com.ibm.xtools.rsarte.webdoc/users-guide/overview.html

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)

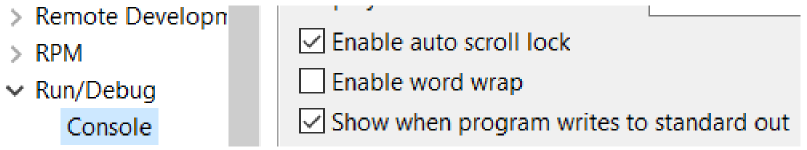


```
29 log.log("Hello! World fr
30 context()->abort();
```

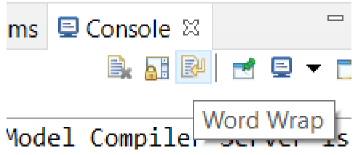
HCL SOFTWARE

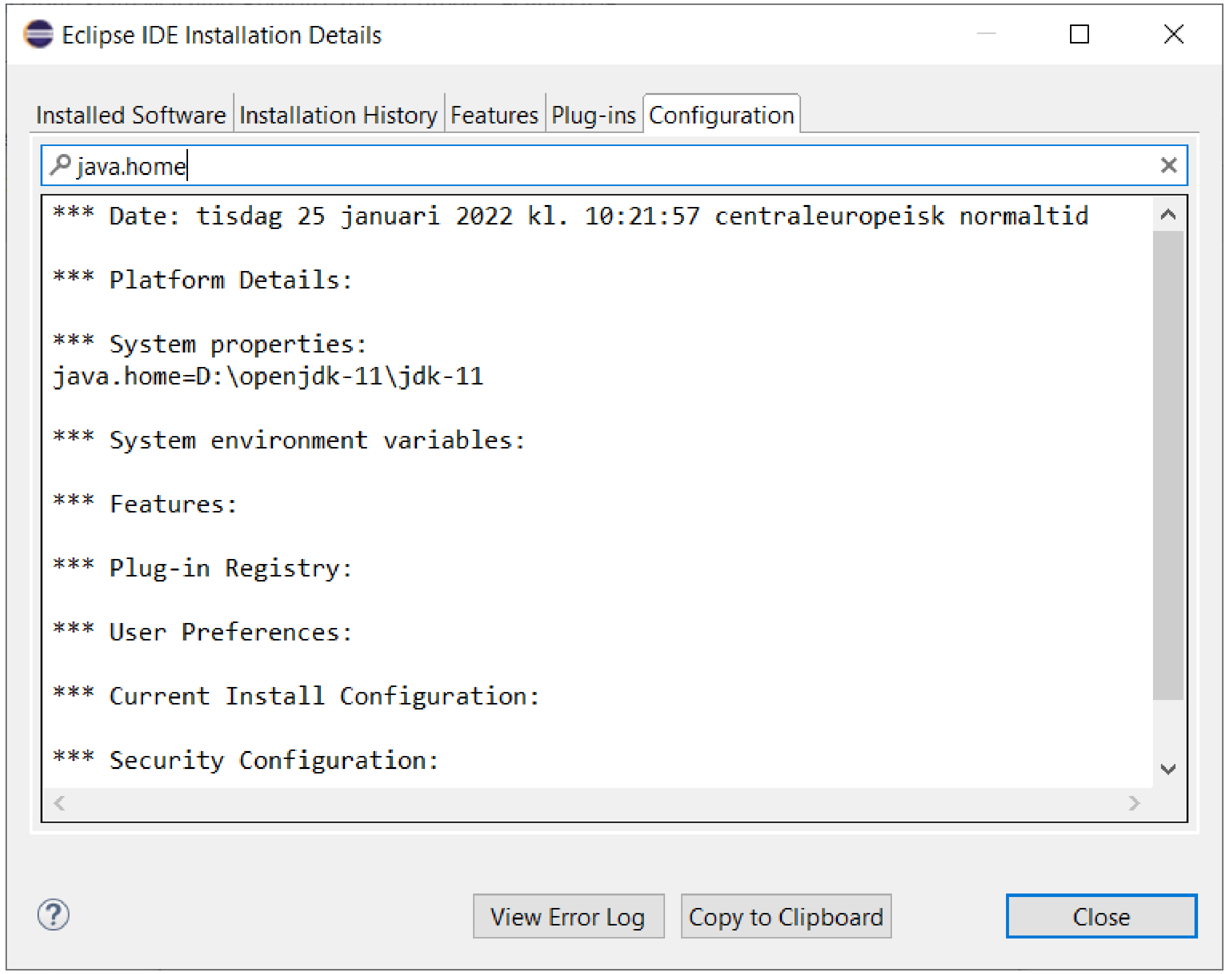
Eclipse 4.20 (2021.06)

- 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



- [✓] Enable auto scroll lock
- [ ] Enable word wrap
- [✓] Show when program writes to standard out

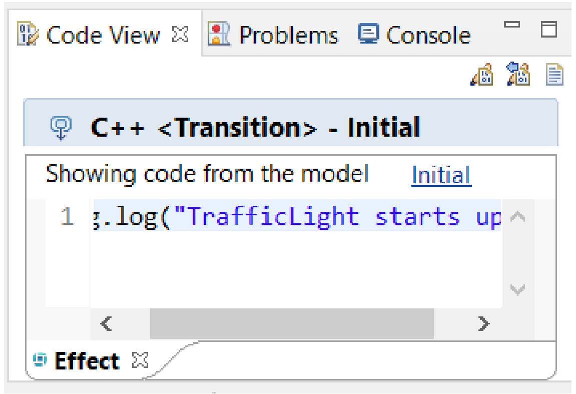




```
*** Date: tisdag 25 januari 2022 kl. 10:21:57 centraleuropeisk normaltid

*** Platform Details:

*** System properties:
java.home=D:\openjdk-11\jdk-11

*** System environment variables:

*** Features:

*** Plug-in Registry:

*** User Preferences:

*** Current Install Configuration:

*** Security Configuration:
```

Eclipse 4.20 (2021.06)

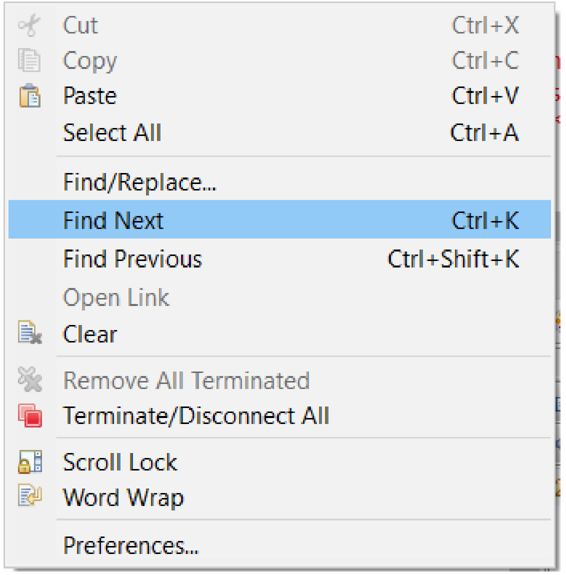
- Horizontal scrolling with Shift + mouse wheel
  - More convenient way of scrolling horizontally if you use a mouse with a scroll wheel
  - Works in all editors (both text editors and diagram editors) and also in views (e.g. the Code view)

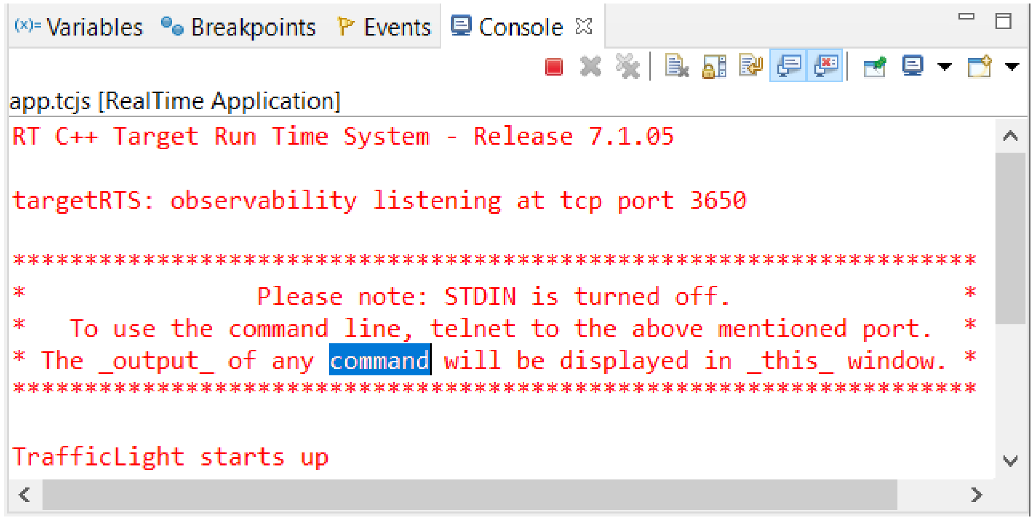


Code View
Problems
Console
C++ <Transition> - Initial
Showing code from the model Initial
Effect

```
1 ;.log("TrafficLight starts up
```

- 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.



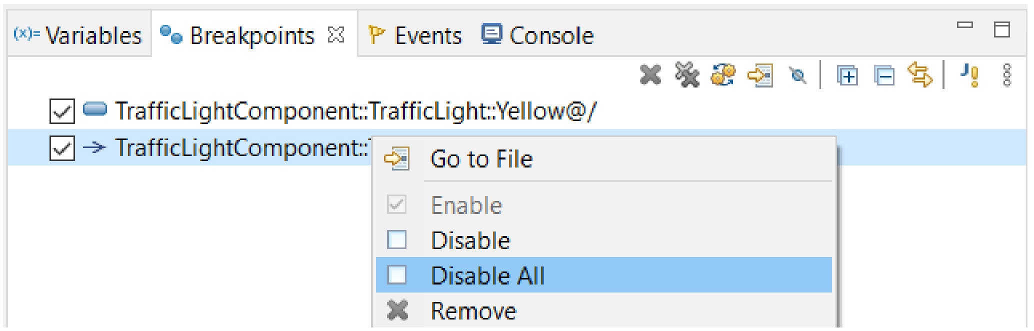


```
app.tcjs [RealTime Application]
RT C++ Target Run Time System - Release 7.1.05

targetRTS: observability listening at tcp port 3650

*********************************************************************
*               Please note: STDIN is turned off.                   *
*   To use the command line, telnet to the above mentioned port.    *
* The _output_ of any command will be displayed in _this_ window.   *
*********************************************************************

TrafficLight starts up
```

HCL SOFTWARE

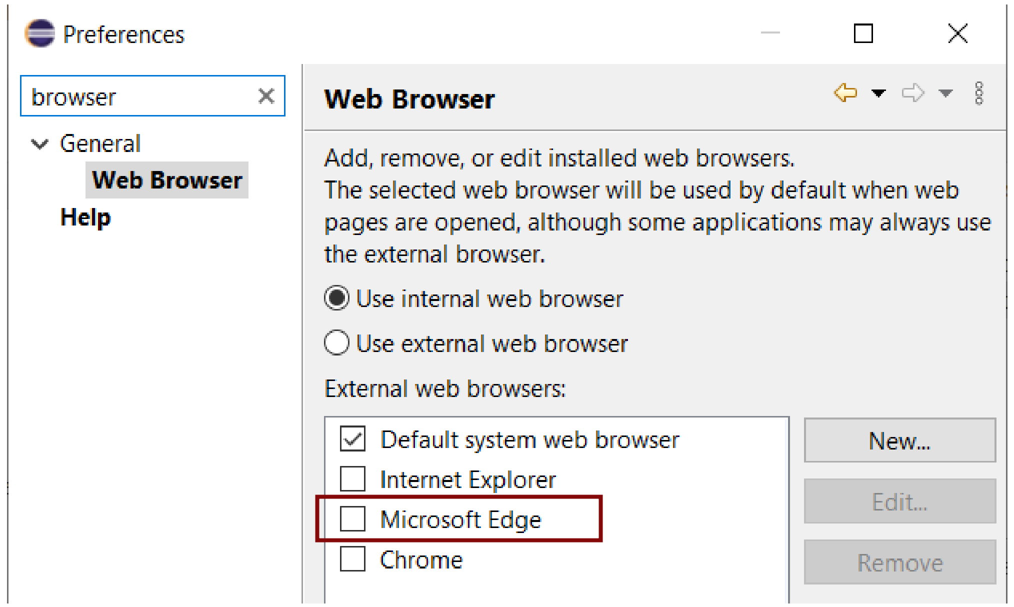
Eclipse 4.20 (2021.06)

- Disable all breakpoints
  - A new context menu command in the Breakpoints view makes this easier



TrafficLightComponent::TrafficLight::Yellow@/
TrafficLightComponent::

- Microsoft Edge is now supported as an external web browser



Web Browser

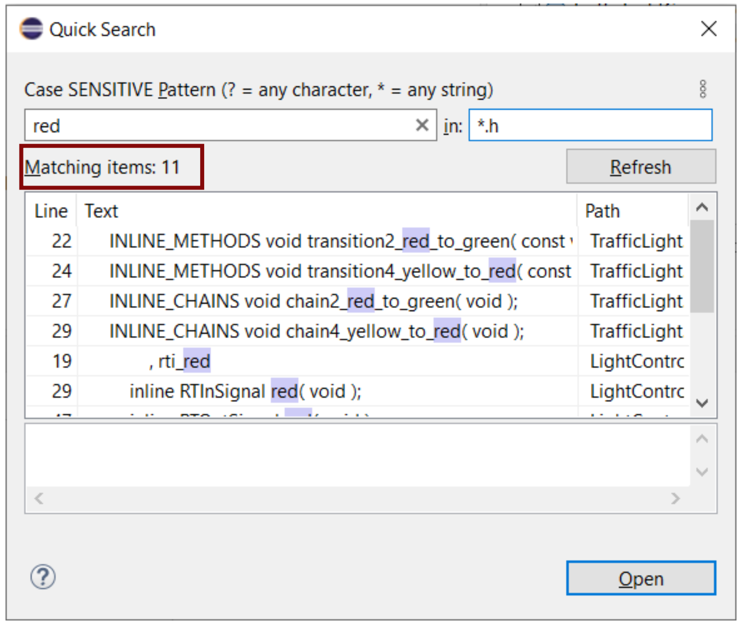
Add, remove, or edit installed web browsers.
The selected web browser will be used by default when web pages are opened, although some applications may always use the external browser.

[✓ Use internal web browser]
Use external web browser

External web browsers:

- [✓] Default system web browser
- [ ] Internet Explorer
- [ ] Microsoft Edge
- [ ] Chrome

- 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



Case SENSITIVE Pattern (? = any character, * = any string)

red in: *.h

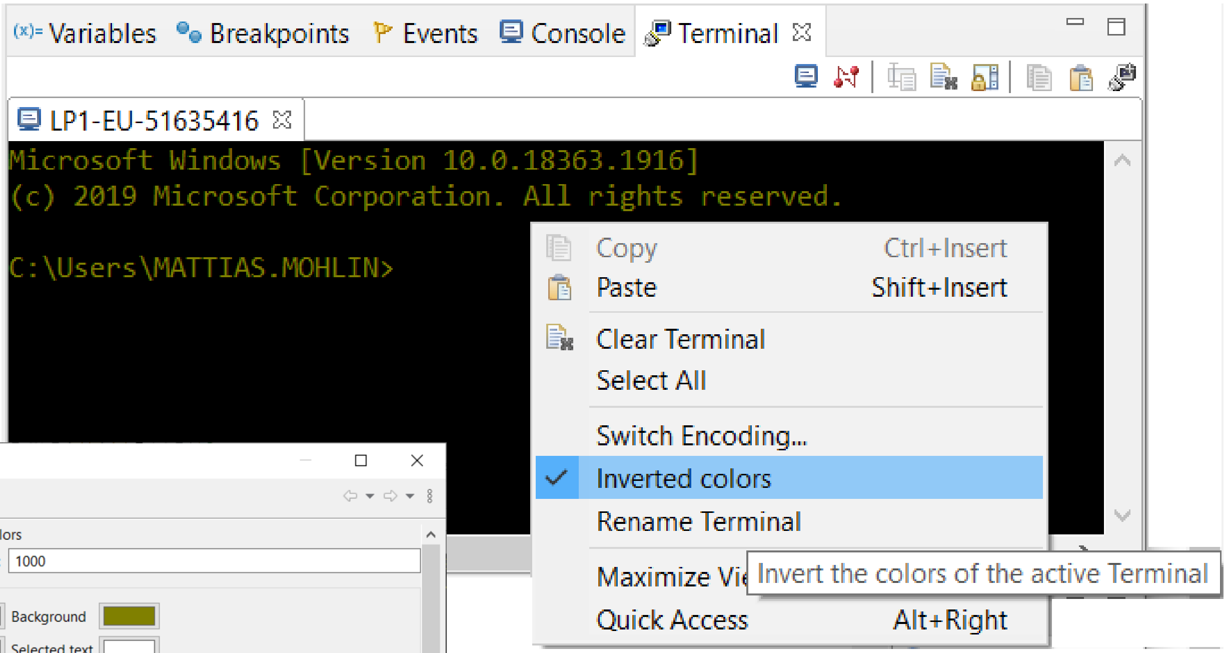
Matching items: 11

Line	Text	Path
22	INLINE_METHODS void transition2_red_to_green(const	TrafficLight
24	INLINE_METHODS void transition4_yellow_to_red(const	TrafficLight
27	INLINE_CHAINS void chain2_red_to_green(void);	TrafficLight
29	INLINE_CHAINS void chain4_yellow_to_red(void);	TrafficLight
19	, rti_red	LightContrc
29	inline RTInSignal red(void);	LightContrc

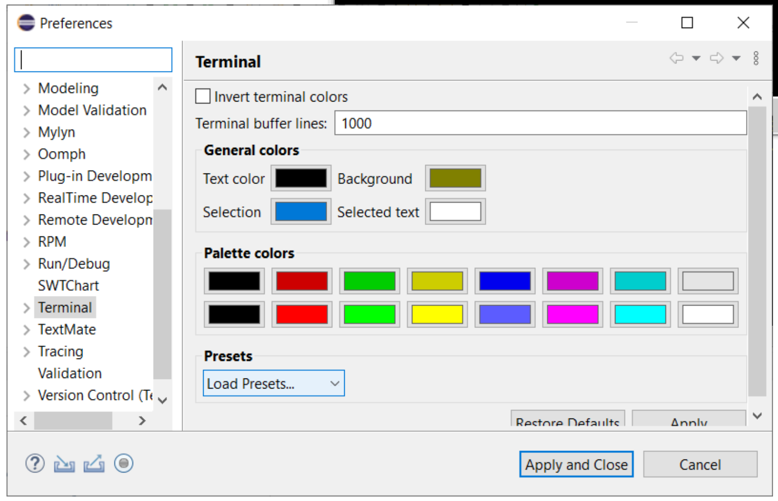
HCL SOFTWARE

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)



```
Microsoft Windows [Version 10.0.18363.1916]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\MATTIAS.MOHLIN>
```



Terminal

☐ Invert terminal colors

Terminal buffer lines: 1000

General colors

Text color ■ Background ■

Selection ■ Selected text □

Palette colors

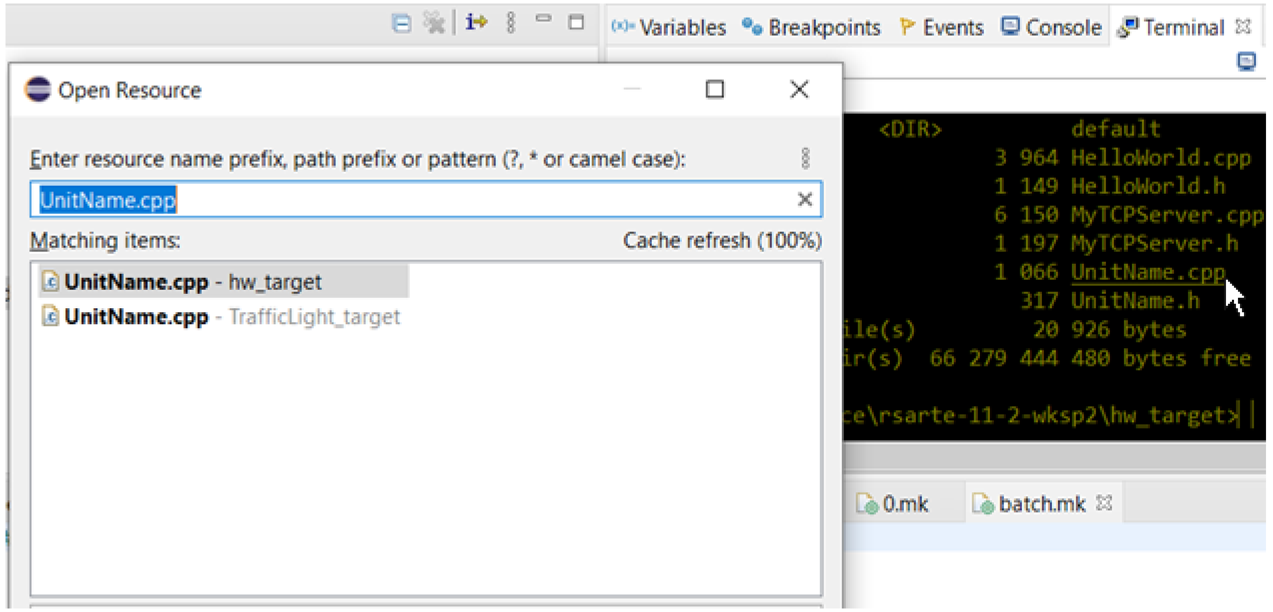
Presets

Load Presets...

HCL SOFTWARE

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



Enter resource name prefix, path prefix or pattern (?, * or camel case):

UnitName.cpp

UnitName.cpp - hw_target

UnitName.cpp - TrafficLight_target

```
 <DIR>          default
         3 964 HelloWorld.cpp
         1 149 HelloWorld.h
         6 150 MyTCPServer.cpp
         1 197 MyTCPServer.h
         1 066 UnitName.cpp
           317 UnitName.h
ile(s)      20 926 bytes
ir(s)  66 279 444 480 bytes free

ce\rsarte-11-2-wksp2\hw_target>
```

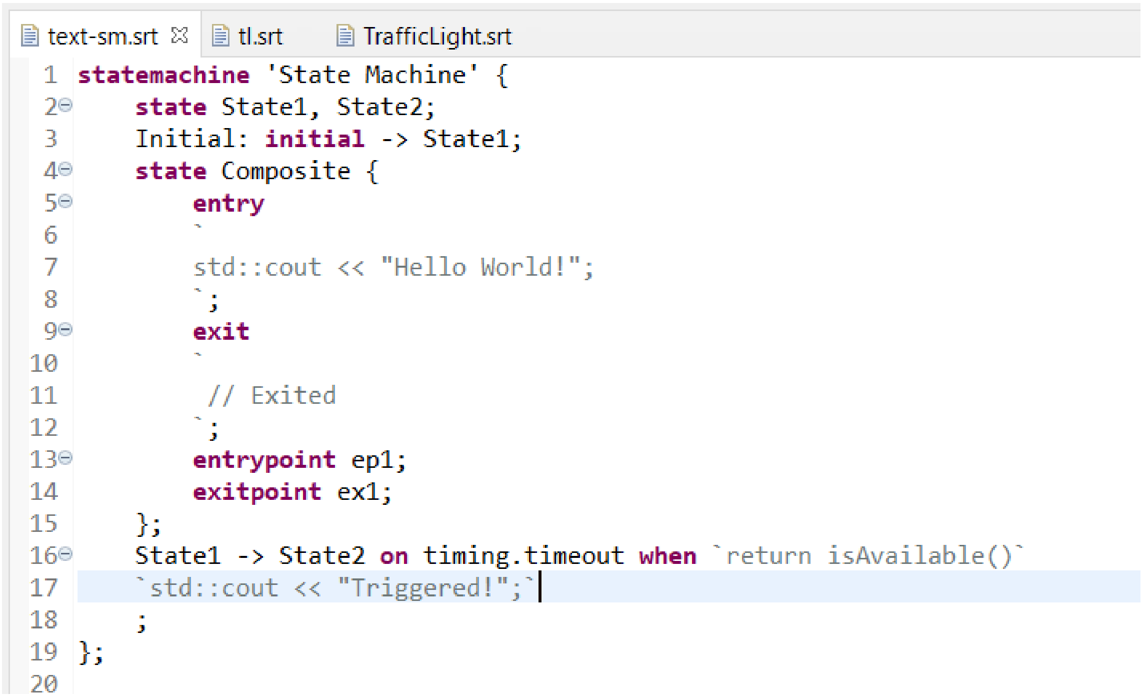
- For more information about CDT improvements see
  https://wiki.eclipse.org/CDT/User/NewIn100
  https://wiki.eclipse.org/CDT/User/NewIn101
  https://wiki.eclipse.org/CDT/User/NewIn102
  https://wiki.eclipse.org/CDT/User/NewIn103

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)

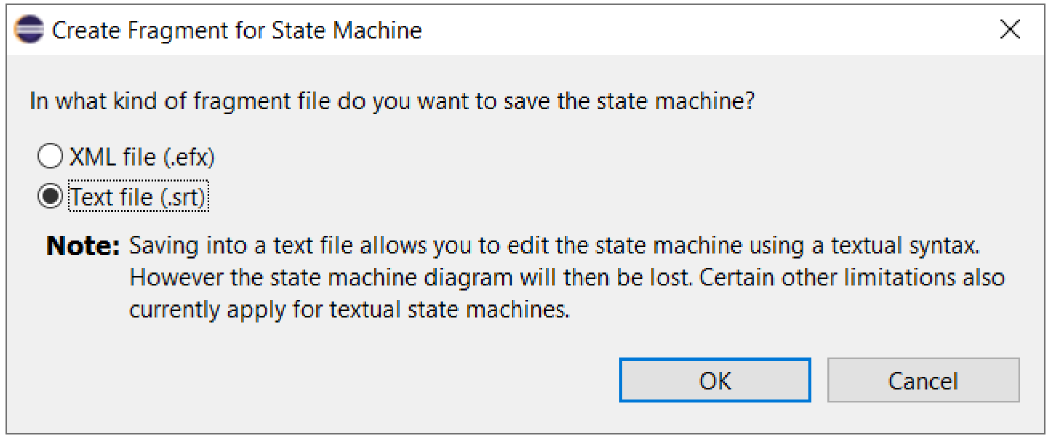
- RTist now supports to define capsule state machines using a textual language
  - The language includes all necessary state machine constructs, including a way to embed C++ code snippets
- This can sometimes be an attractive alternative to using graphical diagrams
  - Certain tasks are faster to perform 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



```
statemachine 'State Machine' {
    state State1, State2;
    Initial: initial -> State1;
    state Composite {
        entry
        `
        std::cout << "Hello World!";
        `;
        exit
        `
         // Exited
        `;
        entrypoint ep1;
        exitpoint ex1;
    };
    State1 -> State2 on timing.timeout when `return isAvailable()`
    `std::cout << "Triggered!";`
    ;
};

```

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)

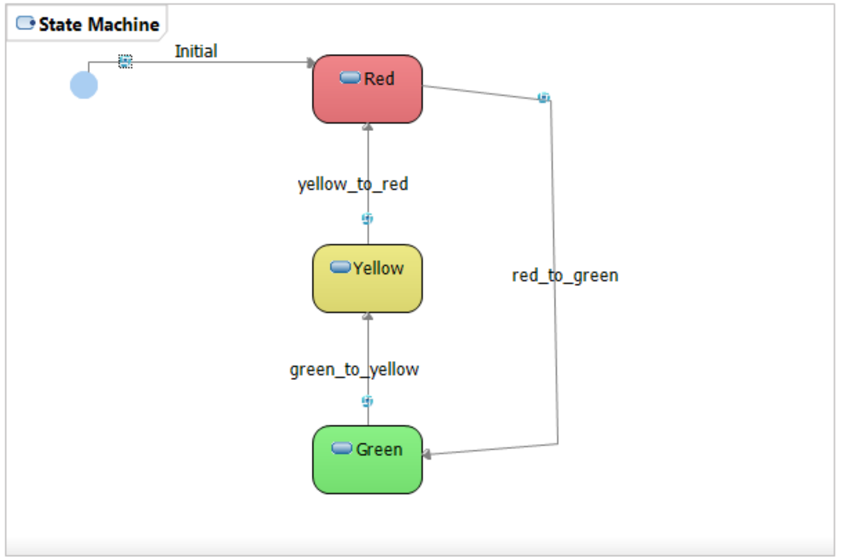

Create Fragment for State Machine

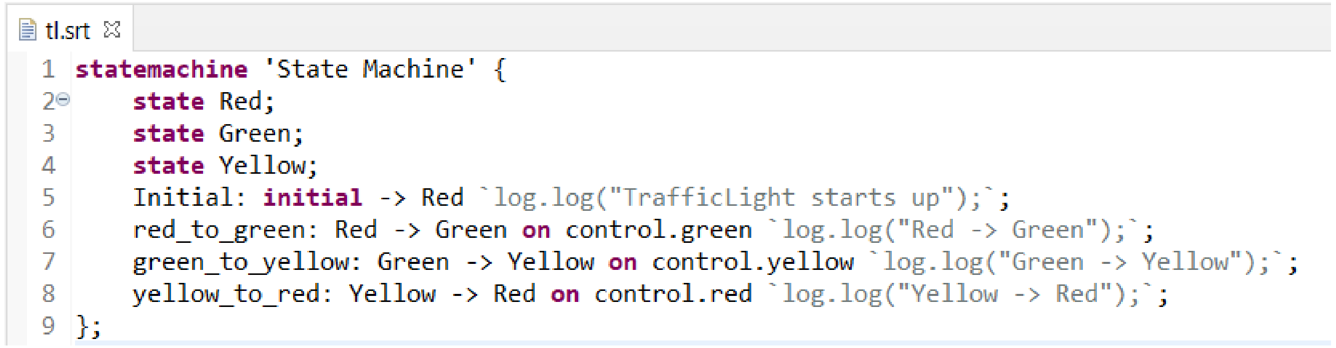
In what kind of fragment file do you want to save the state machine?

( ) XML file (.efx)
[✓ Text file (.srt)]

Note: Saving into a text file allows you to edit the state machine using a textual syntax. However the state machine diagram will then be lost. Certain other limitations also currently apply for textual state machines.

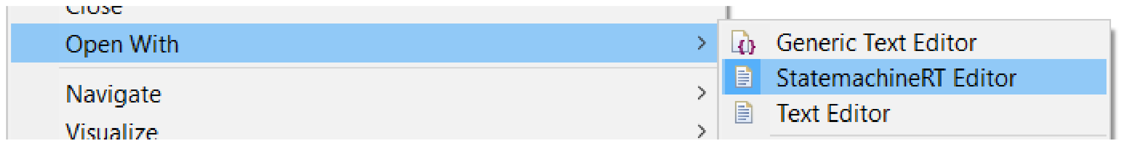
OK | Cancel



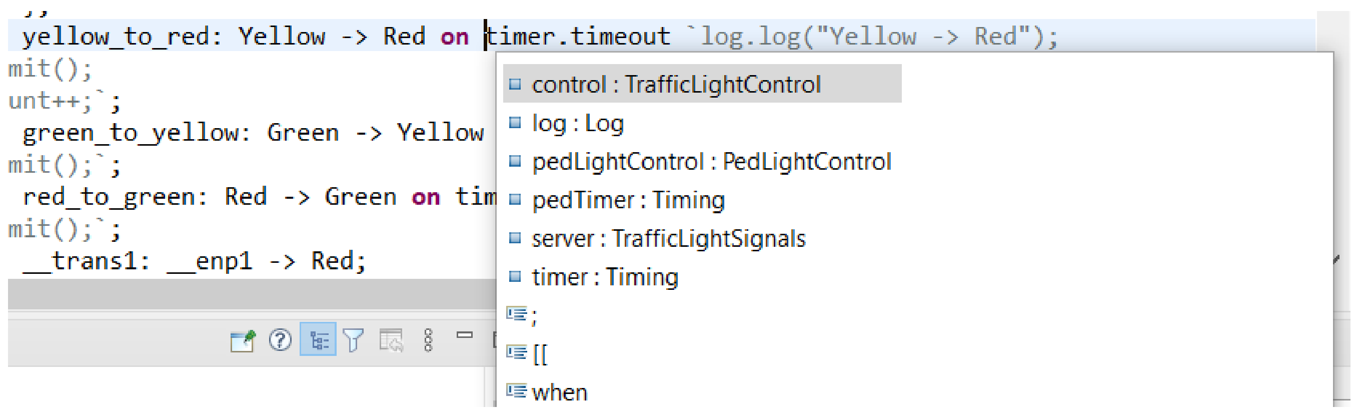

tl.srt

```
statemachine 'State Machine' {
    state Red;
    state Green;
    state Yellow;
    Initial: initial -> Red `log.log("TrafficLight starts up");`;
    red_to_green: Red -> Green on control.green `log.log("Red -> Green");`;
    green_to_yellow: Green -> Yellow on control.yellow `log.log("Green -> Yellow");`;
    yellow_to_red: Yellow -> Red on control.red `log.log("Yellow -> Red");`;
};
```

Textual State Machines (3/4)



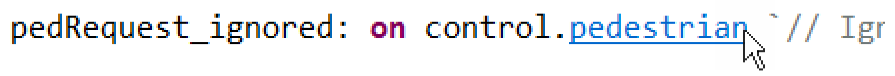
- Textual state machines can be edited in any text editor, but are best edited in the "StatemachineRT" text editor provided in RTist (associated in Eclipse with the .srt file extension)



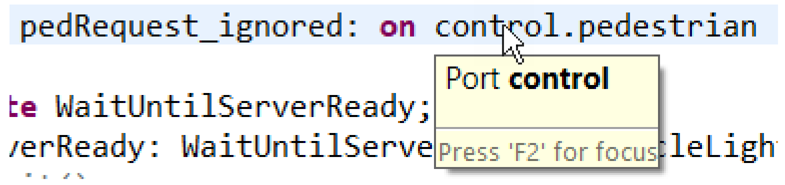
```
 yellow_to_red: Yellow -> Red on timer.timeout `log.log("Yellow -> Red");
mit();
unt++;`;
 green_to_yellow: Green -> Yellow
mit();`;
 red_to_green: Red -> Green on tim
mit();`;
 __trans1: __enp1 -> Red;
```

control : TrafficLightControl
log : Log
pedLightControl : PedLightControl
pedTimer : Timing
server : TrafficLightSignals
timer : Timing
;
[[
when

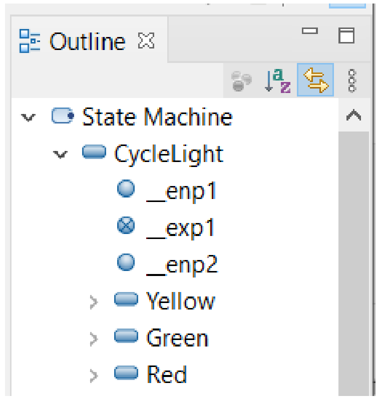
  - Ctrl + space content assist (provides both code templates for creating new elements, and automatic completion of names)
  - Ctrl + click for navigating from a name reference to the corresponding definition (in the Project Explorer, or an in the same or a different .srt file)



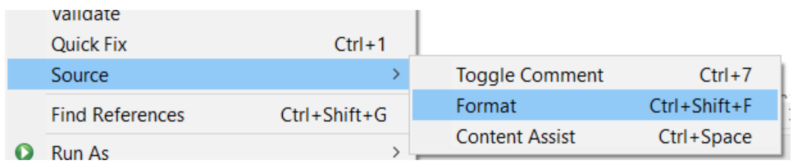
```
pedRequest_ignored: on control.pedestrian `// Igr
```



```
pedRequest_ignored: on control.pedestrian
te WaitUntilServerReady;
/erReady: WaitUntilServe            leLigh
```



  - 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

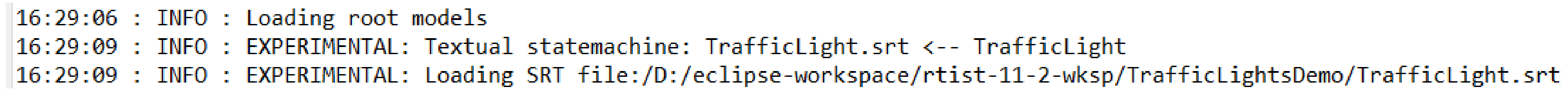


HCL SOFTWARE

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



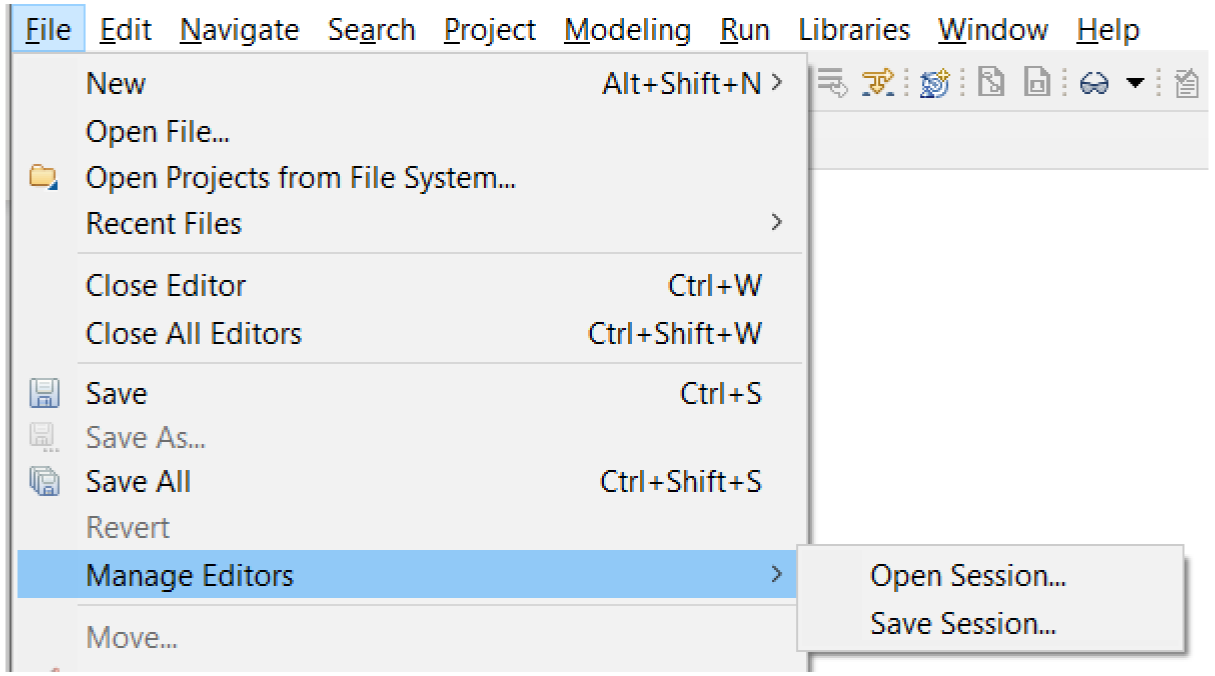
```
16:29:06 : INFO : Loading root models
16:29:09 : INFO : EXPERIMENTAL: Textual statemachine: TrafficLight.srt <-- TrafficLight
16:29:09 : INFO : EXPERIMENTAL: Loading SRT file:/D:/eclipse-workspace/rtist-11-2-wksp/TrafficLightsDemo/TrafficLight.srt
```

- 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.

HCL SOFTWARE

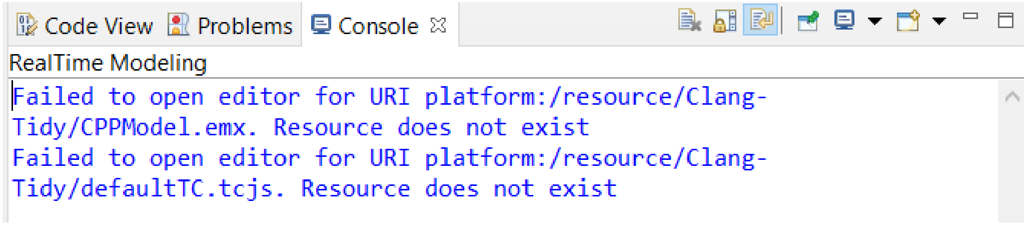
Manage Editors

- 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

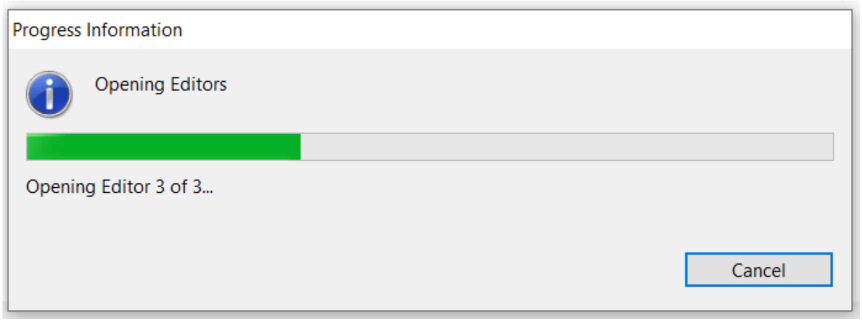


```
Failed to open editor for URI platform:/resource/Clang-
Tidy/CPPModel.emx. Resource does not exist
Failed to open editor for URI platform:/resource/Clang-
Tidy/defaultTC.tcjs. Resource does not exist
```

HCL SOFTWARE

Automatic Restoration of Modeling Editors

- A new preference can be set to attempt to automatically restore previously open modeling editors after a restart of Eclipse
  - **Modeling - Automatically restore modeling editors on startup**

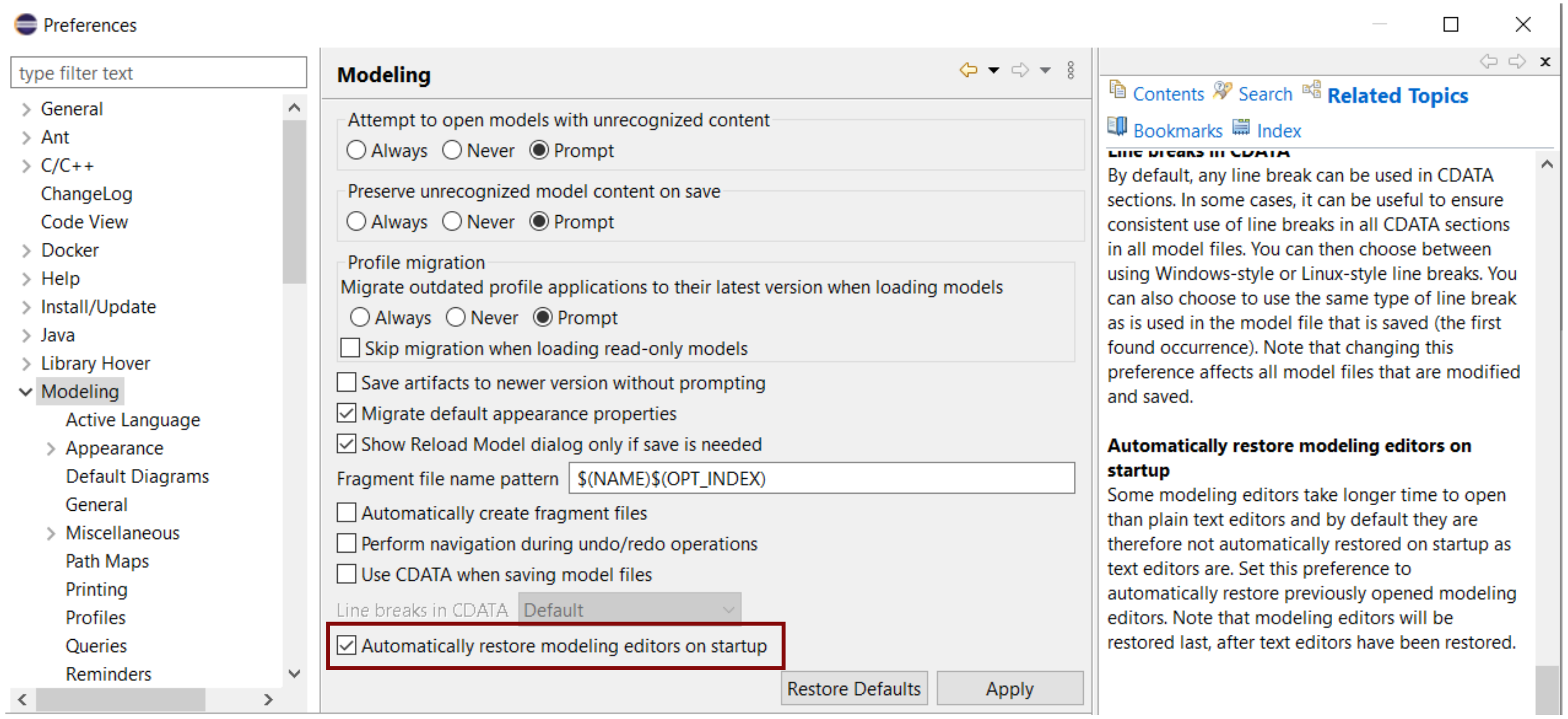


Progress Information

Opening Editors

Opening Editor 3 of 3...

Cancel



Preferences

type filter text

General
Ant
C/C++
ChangeLog
Code View
Docker
Help
Install/Update
Java
Library Hover
Modeling
  Active Language
  Appearance
  Default Diagrams
  General
  Miscellaneous
  Path Maps
  Printing
  Profiles
  Queries
  Reminders

Modeling

Attempt to open models with unrecognized content
Always Never [✓ Prompt]

Preserve unrecognized model content on save
Always Never [✓ Prompt]

Profile migration
Migrate outdated profile applications to their latest version when loading models
Always Never [✓ Prompt]
☐ Skip migration when loading read-only models

☐ Save artifacts to newer version without prompting
☑ Migrate default appearance properties
☑ Show Reload Model dialog only if save is needed
Fragment file name pattern: $(NAME)$(OPT_INDEX)
☐ Automatically create fragment files
☐ Perform navigation during undo/redo operations
☐ Use CDATA when saving model files
Line breaks in CDATA: Default
☑ Automatically restore modeling editors on startup

Restore Defaults | Apply

Contents | Search | **Related Topics**
Bookmarks | Index

Line breaks in CDATA
By default, any line break can be used in CDATA sections. In some cases, it can be useful to ensure consistent use of line breaks in all CDATA sections in all model files. You can then choose between using Windows-style or Linux-style line breaks. You can also choose to use the same type of line break as is used in the model file that is saved (the first found occurrence). Note that changing this preference affects all model files that are modified and saved.

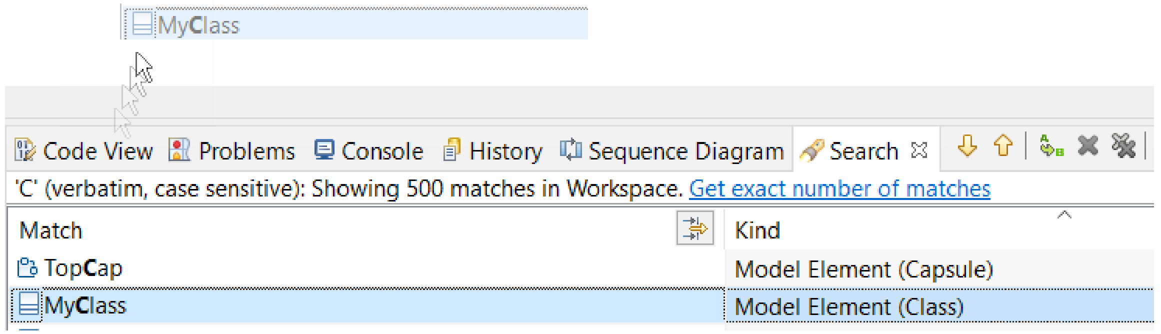
Automatically restore modeling editors on startup
Some modeling editors take longer time to open than plain text editors and by default they are therefore not automatically restored on startup as text editors are. Set this preference to automatically restore previously opened modeling editors. Note that modeling editors will be restored last, after text editors have been restored.

- This is an extension to the built-in restoration of editors in Eclipse
  - Modeling editors usually take longer time to open than regular Eclipse editors
  - Modeling editors are placed after the regular Eclipse editors in the workbench editor area

HCL SOFTWARE

Drag/drop Support from More Views

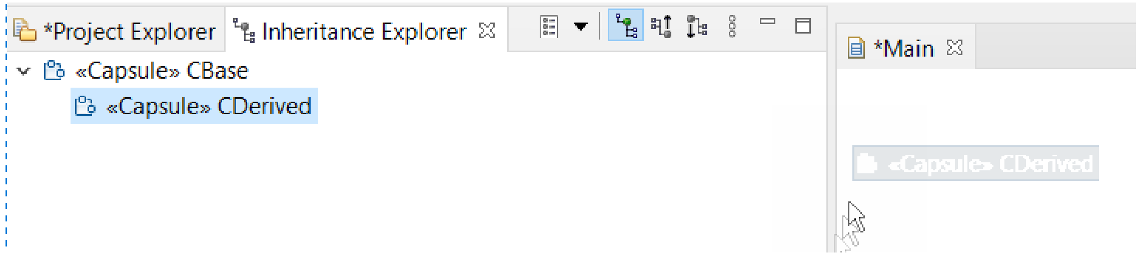
- Now it's possible to drag/drop elements from other views than the Project Explorer to diagrams
  - Search view
  - Inheritance Explorer



'C' (verbatim, case sensitive): Showing 500 matches in Workspace. Get exact number of matches

Match	Kind
TopCap	Model Element (Capsule)
MyClass	Model Element (Class)

- 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

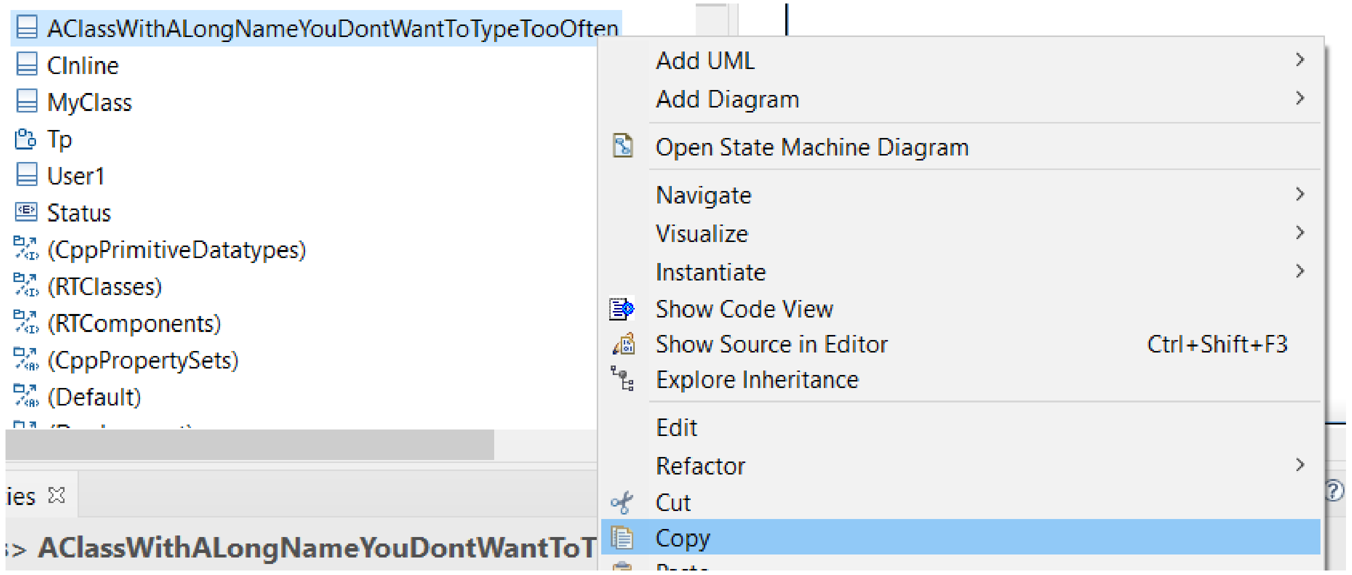


- Known limitation: Dragging a capsule to a composite structure diagram in order to create a part for it doesn't work
  - In this particular case, it's still necessary to drag from the Project Explorer

HCL SOFTWARE

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

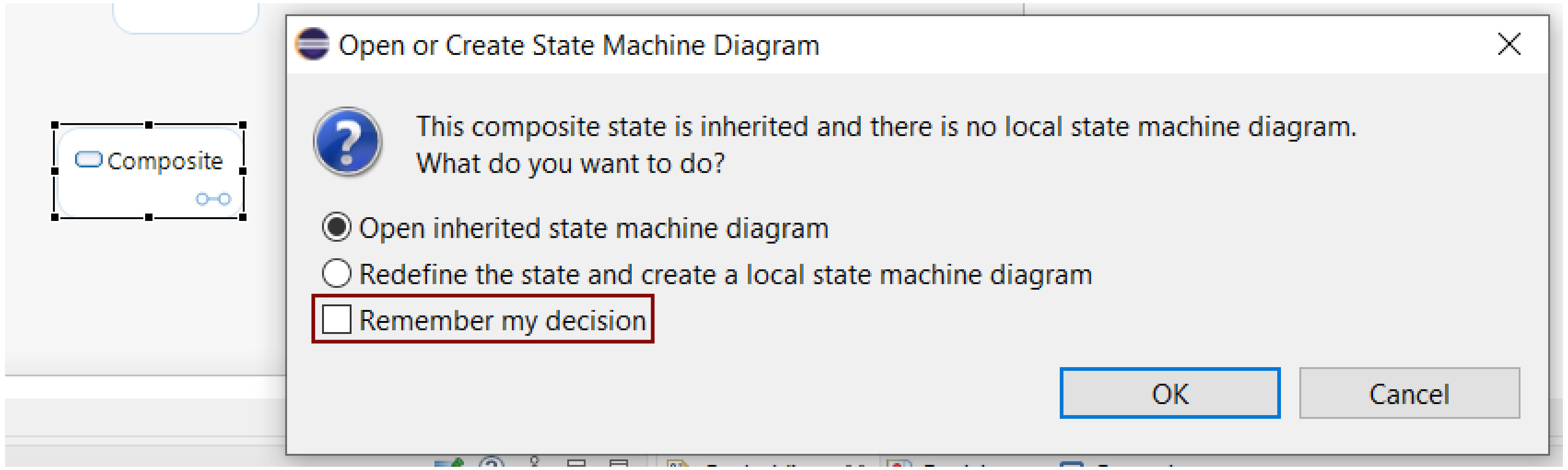

AClassWithALongNameYouDontWantToTypeTooOften
CInline
MyClass
Tp
User1
Status
(CppPrimitiveDatatypes)
(RTClasses)
(RTComponents)
(CppPropertySets)
(Default)

> AClassWithALongNameYouDontWantToT

HCL SOFTWARE

Navigating into Inherited Composite States

- The choice made in the "Open or Create State Machine Diagram" dialog can now be remembered
  - Stored in a new preference **RealTime Development - Diagrams - State Chart - When going inside inherited composite state**
  - For example useful when you just browse around in a model and don't want to modify it by mistake



Open or Create State Machine Diagram

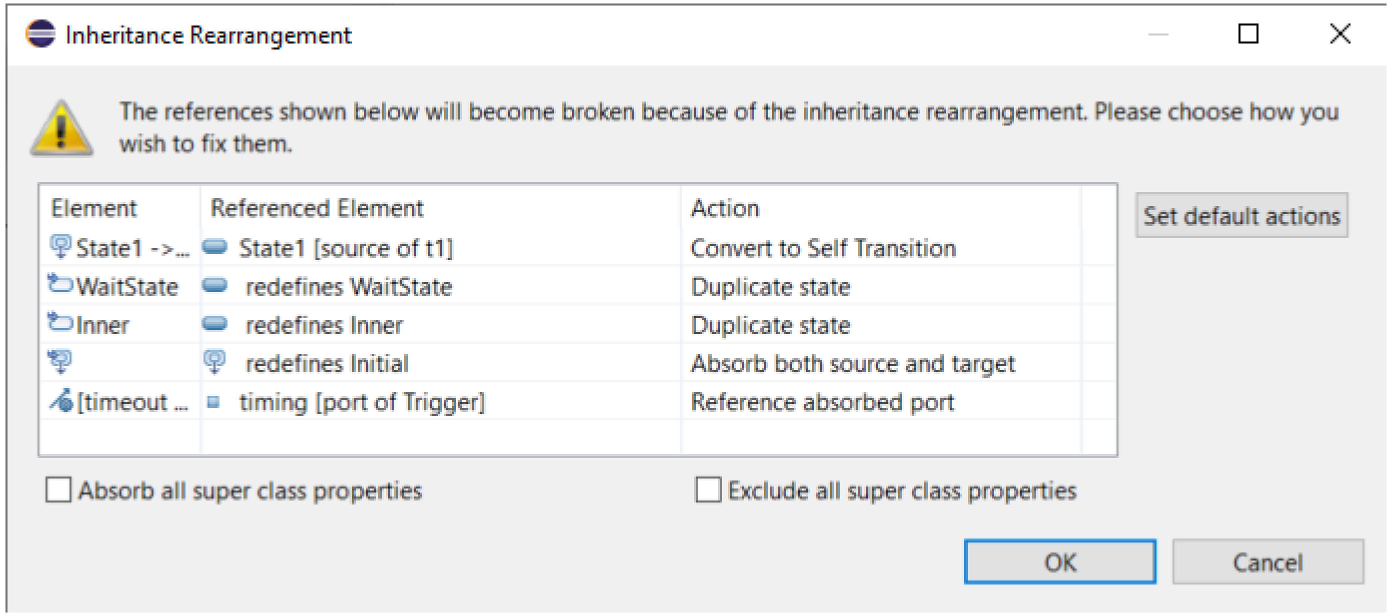
This composite state is inherited and there is no local state machine diagram.
What do you want to do?

[✓ Open inherited state machine diagram]
Redefine the state and create a local state machine diagram
☐ Remember my decision

HCL SOFTWARE

Improved Inheritance Rearrangement Dialog (1/2)

- When deleting or redirecting a capsule generalization, referenced elements from the previous super capsule may no longer be accessible after the refactoring
- Previously, referenced elements were always absorbed (i.e. copied) into the subcapsule to avoid such references from becoming broken (and prevent loss of data)
  - This happened even for the cases when the referenced elements were still accessible after the inheritance rearrangement (i.e. the case when the new super capsule also inherits from the old super capsule).
- Now the Inheritance Rearrangement dialog is improved for more flexibility:
  - Affected references, and the referenced elements, are listed so you can know how the inheritance rearrangement will affect the subcapsule before you decide to proceed
  - For each reference you can choose which action to take for preventing it from becoming unbound



Inheritance Rearrangement

The references shown below will become broken because of the inheritance rearrangement. Please choose how you wish to fix them.

Element	Referenced Element	Action
State1 ->...	State1 [source of t1]	Convert to Self Transition
WaitState	redefines WaitState	Duplicate state
Inner	redefines Inner	Duplicate state
	redefines Initial	Absorb both source and target
[timeout ...	timing [port of Trigger]	Reference absorbed port

Set default actions

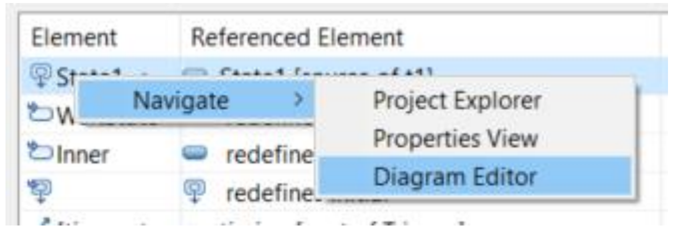
☐ Absorb all super class properties ☐ Exclude all super class properties

OK Cancel

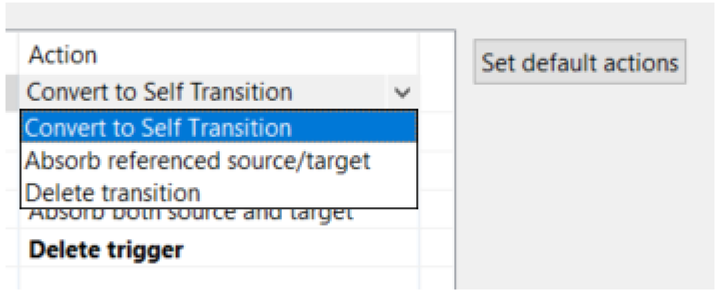
HCL SOFTWARE

Improved Inheritance Rearrangement Dialog (2/2)

- You can navigate to find a reference in Diagram Editor, Project Explorer or Properties view
  - Double-click to navigate to Project Explorer
  - Right-click to get context menu with all navigation options
  - Navigation helps you decide the right action for each reference



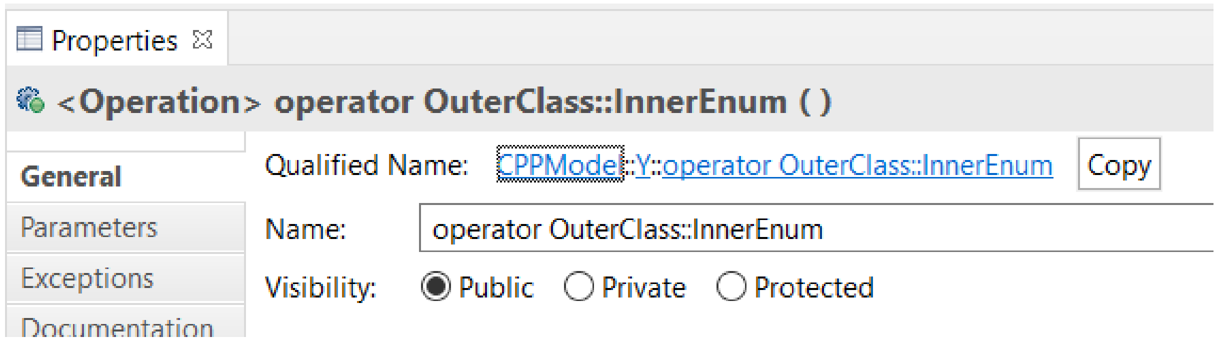
- Default actions are chosen so that no information will be lost in the refactoring
  - Override default actions using the drop-down menu in the Action column
  - Overridden actions are marked in boldface
  - Click the **Set default actions** button to restore all actions to the default ones


Action
Convert to Self Transition
Convert to Self Transition
Absorb referenced source/target
Delete transition
Absorb both source and target
Delete trigger
Set default actions

- The checkboxes for absorbing and excluding super class properties are still available and works as before
  - Note that if you choose to absorb all super class properties, you cannot override any actions in the table

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 instantations are now also supported



<Operation> operator OuterClass::InnerEnum ( )

Qualified Name: CPPModel::Y::operator OuterClass::InnerEnum

Name: operator OuterClass::InnerEnum

Visibility: [✓ Public] Private Protected

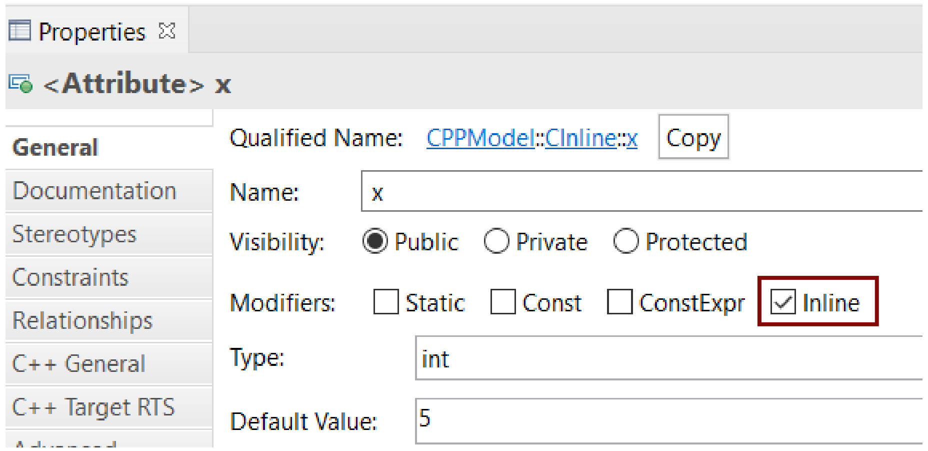
HCL SOFTWARE

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



Properties

<Attribute> x

General | Documentation | Stereotypes | Constraints | Relationships | C++ General | C++ Target RTS

Qualified Name: CPPModel::CInline::x Copy

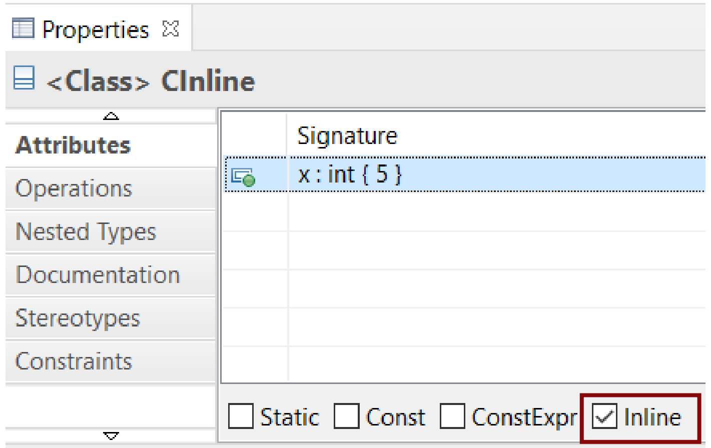
Name: x

Visibility: [✓ Public] Private Protected

Modifiers: Static Const ConstExpr [✓ Inline]

Type: int

Default Value: 5



Properties

<Class> CInline

Attributes | Operations | Nested Types | Documentation | Stereotypes | Constraints

	Signature
	x : int { 5 }

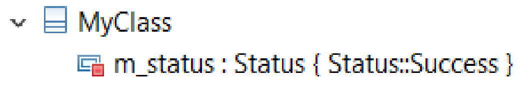
Static Const ConstExpr [✓ Inline]

```
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

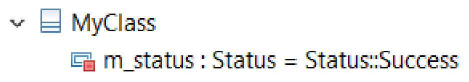


MyClass
m_status : Status { Status::Success }

Initializer Kind: brace

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

- Equal / assignment initialization

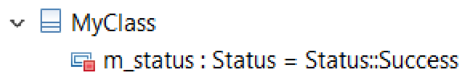


MyClass
m_status : Status = Status::Success

Initializer Kind: equal
Initializer Kind: assignment

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

- Constructor initialization



MyClass
m_status : Status = Status::Success

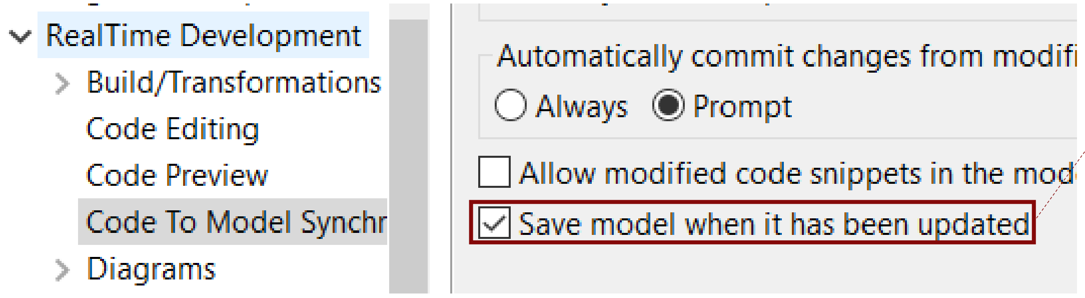
Initializer Kind: constructor

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

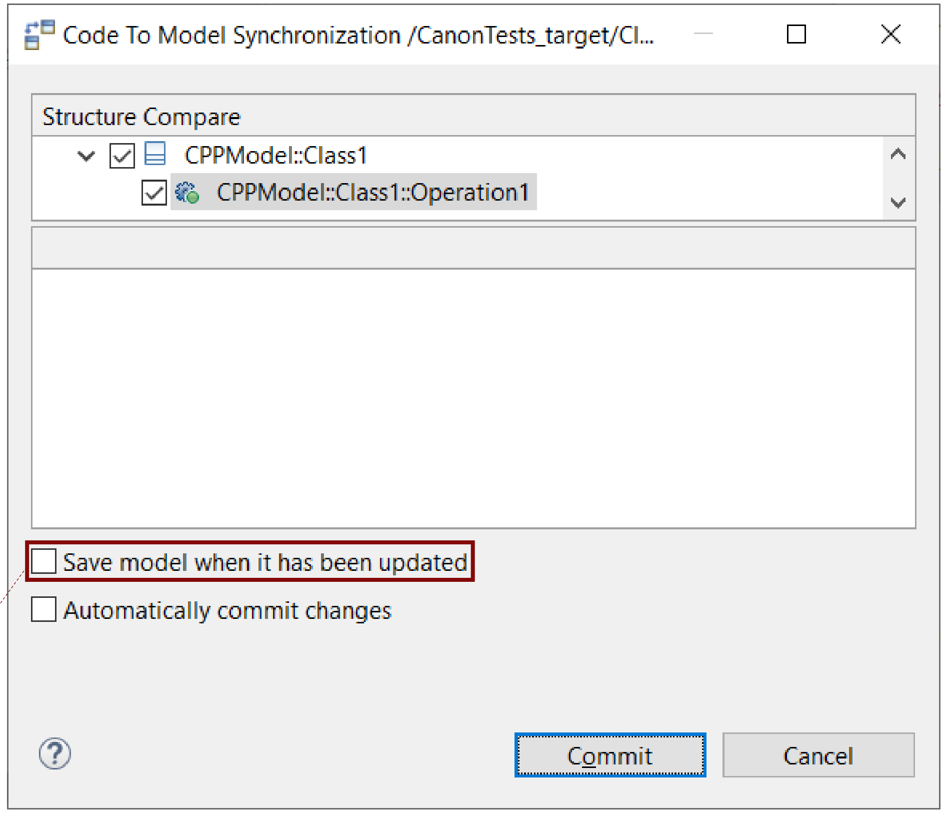
HCL SOFTWARE

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



Automatically commit changes from modifi
( ) Always [✓ Prompt]
☐ Allow modified code snippets in the mod
☑ Save model when it has been updated



Code To Model Synchronization /CanonTests_target/Cl...

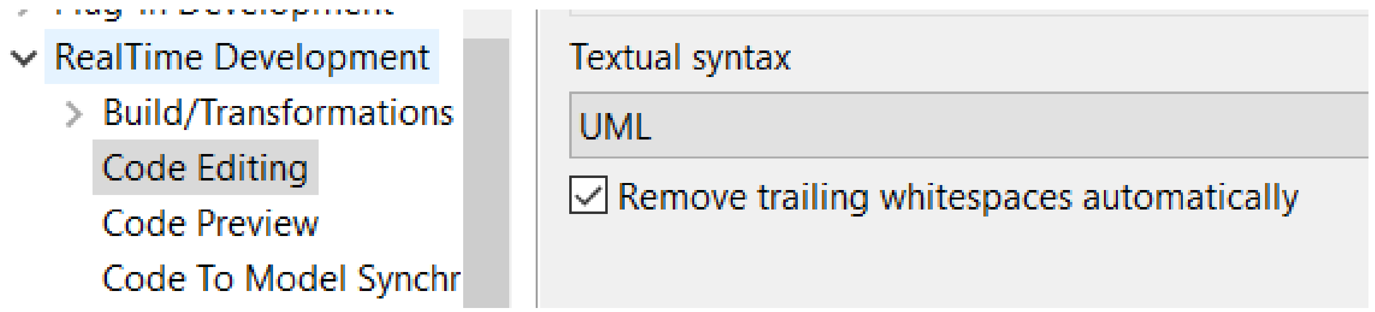
Structure Compare
☑ CPPModel::Class1
☑ CPPModel::Class1::Operation1

☐ Save model when it has been updated
☐ Automatically commit changes

HCL SOFTWARE

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.



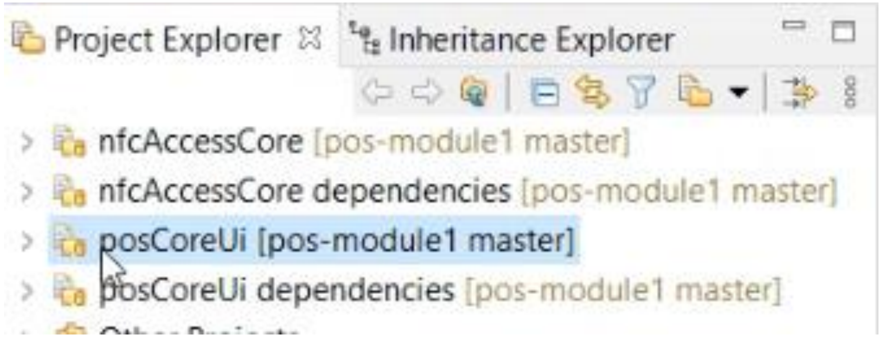
Textual syntax

UML

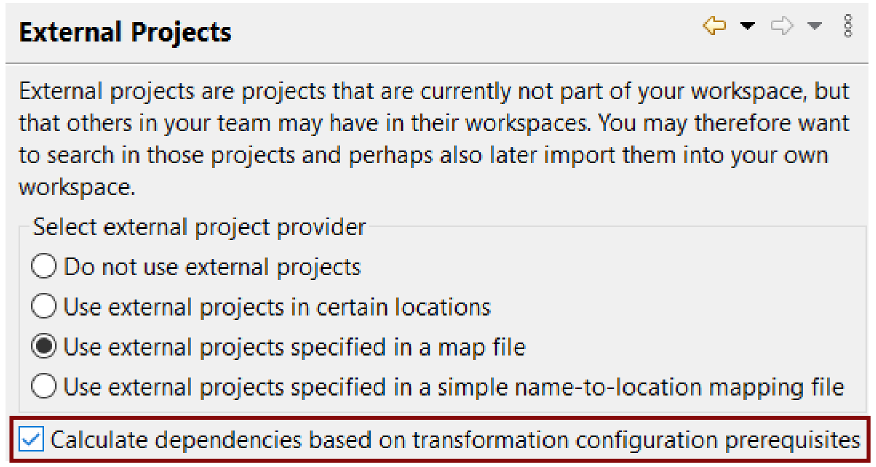
☑ Remove trailing whitespaces automatically

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



Project Explorer
Inheritance Explorer
nfcAccessCore [pos-module1 master]
nfcAccessCore dependencies [pos-module1 master]
posCoreUi [pos-module1 master]
posCoreUi dependencies [pos-module1 master]


External Projects

External projects are projects that are currently not part of your workspace, but that others in your team may have in their workspaces. You may therefore want to search in those projects and perhaps also later import them into your own workspace.

Select external project provider

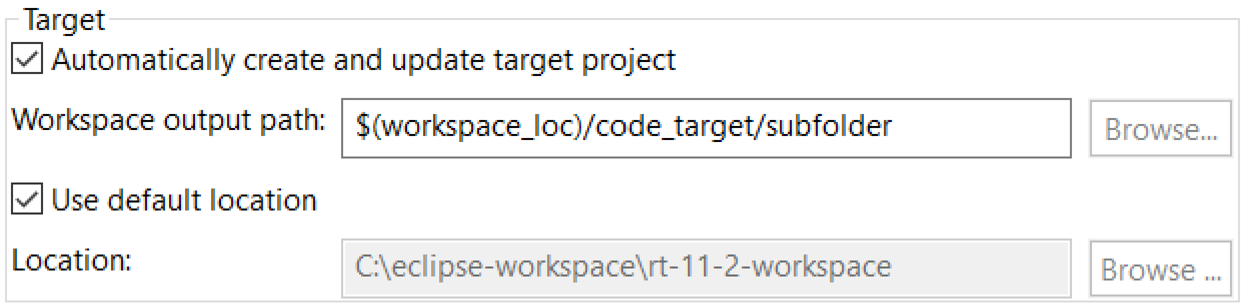
- ○ Do not use external projects
- ○ Use external projects in certain locations
- ● Use external projects specified in a map file
- ○ Use external projects specified in a simple name-to-location mapping file

☑ Calculate dependencies based on transformation configuration prerequisites

HCL SOFTWARE

Transformation Configurations (1/2)

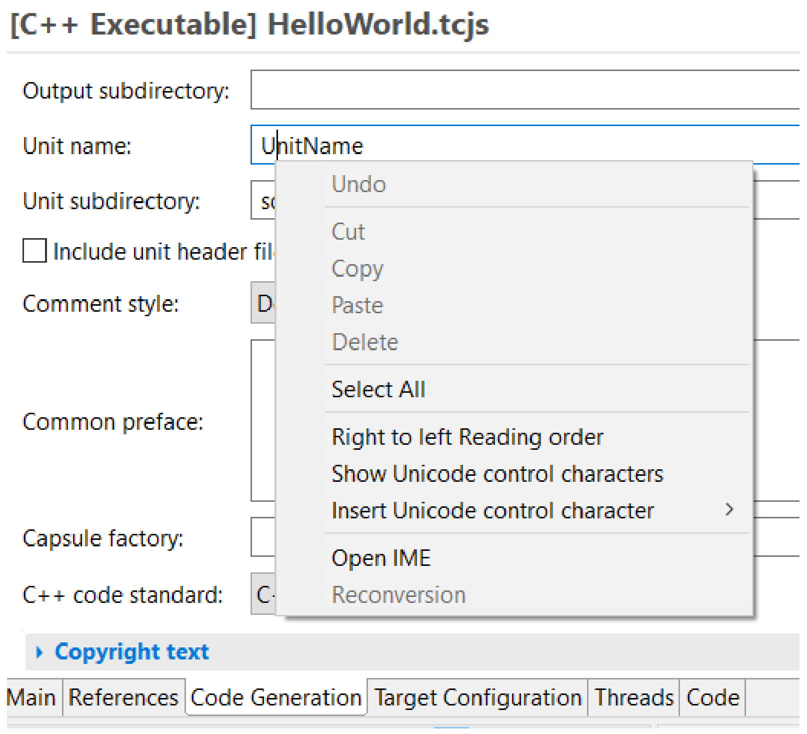
- The ”Workspace output path” property now allows for more flexibility
  - An absolute path can be used
  - Subfolders are supported
  - If the variable $(TCONFIG_NAME) is used for this property in an inherited TC, it’s now correctly resolved with the name of the local TC
- The standard Eclipse variable $(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: $(workspace_loc)/code_target/subfolder
☑ Use default location
Location: C:\eclipse-workspace\rt-11-2-workspace

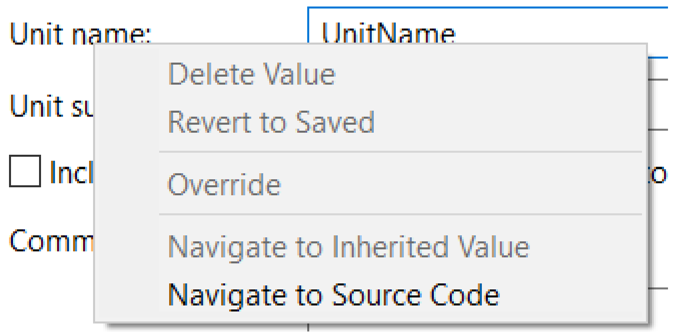
HCL SOFTWARE

Transformation Configurations (2/2)

- The TC editor now provides the standard context menu for text fields
  - Right-click on the property label to bring up the TC editor specific context menu



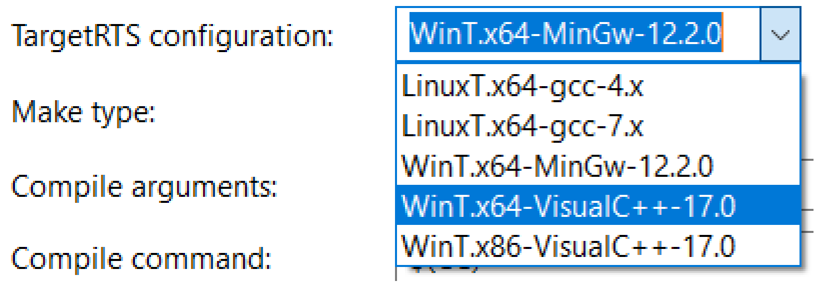
standard context menu for text field



TC editor specific context menu for property label

Uplift to Newer Compiler Versions

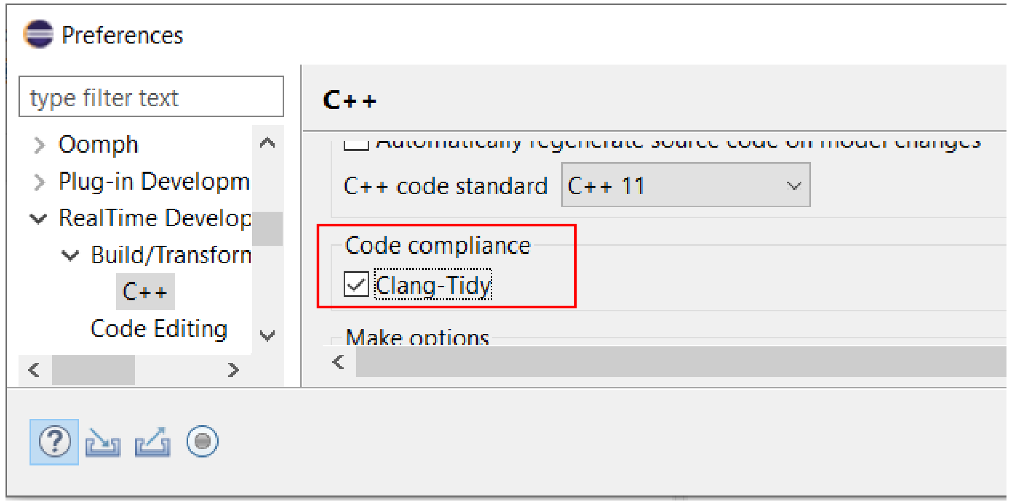
- The Visual Studio 2022 C++ compiler (ver 17.0) can now be directly used with RTist
  - A target configuration for this compiler is available
  - Prebuilt libraries for this compiler are included (TargetRTS, Connexis, LibTCPServer)
  - Replaces the old Visual Studio 2017 (ver 15.0) target configurations and libraries
  - Both 64 and 32 bit versions are available
- The MinGW 12.2 C++ compiler 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 MinGw 8.1 target configurations and libraries


TargetRTS configuration: WinT.x64-MinGw-12.2.0
Make type:
Compile arguments:
Compile command:

LinuxT.x64-gcc-4.x
LinuxT.x64-gcc-7.x
WinT.x64-MinGw-12.2.0
WinT.x64-VisualC++-17.0
WinT.x86-VisualC++-17.0

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.


Preferences
type filter text
Oomph
Plug-in Developm
RealTime Develop
Build/Transforn
C++
Code Editing
C++
Automatically regenerate source code on model changes
C++ code standard C++ 11
Code compliance
[✓] Clang-Tidy
Make options

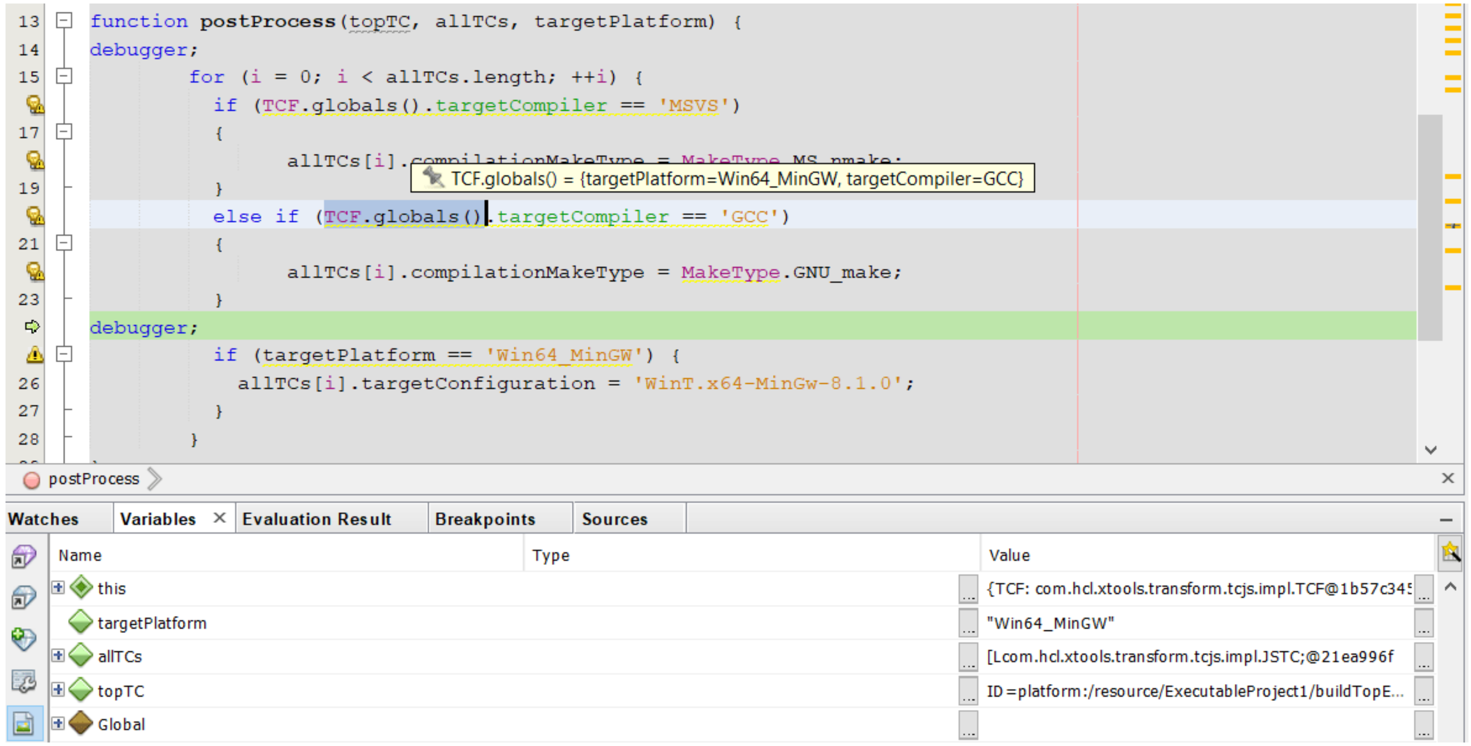
```
inline RTTypedValue_MyClass( const MyClass & rtg_value /* NOLINT(google-explicit-constructor) */ )
```

- When the C++ language standard is C++ 17 or later, the Clang-Tidy rule **misc-unused-parameters** is now suppressed by means of a standard C++ attribute `[[maybe_unused]]` rather than a NOLINT comment.

```
static void rtg_C1_init( [[maybe_unused]] const RTObject_class * type, [[maybe_unused]] C1 * target )
```

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



```
function postProcess(topTC, allTCs, targetPlatform) {
debugger;
        for (i = 0; i < allTCs.length; ++i) {
          if (TCF.globals().targetCompiler == 'MSVS')
          {
                allTCs[i].compilationMakeType = MakeType.MS_nmake;
          }
          else if (TCF.globals().targetCompiler == 'GCC')
          {
                allTCs[i].compilationMakeType = MakeType.GNU_make;
          }
debugger;
          if (targetPlatform == 'Win64_MinGW') {
            allTCs[i].targetConfiguration = 'WinT.x64-MinGw-8.1.0';
          }
        }
```

TCF.globals() = {targetPlatform=Win64_MinGW, targetCompiler=GCC}

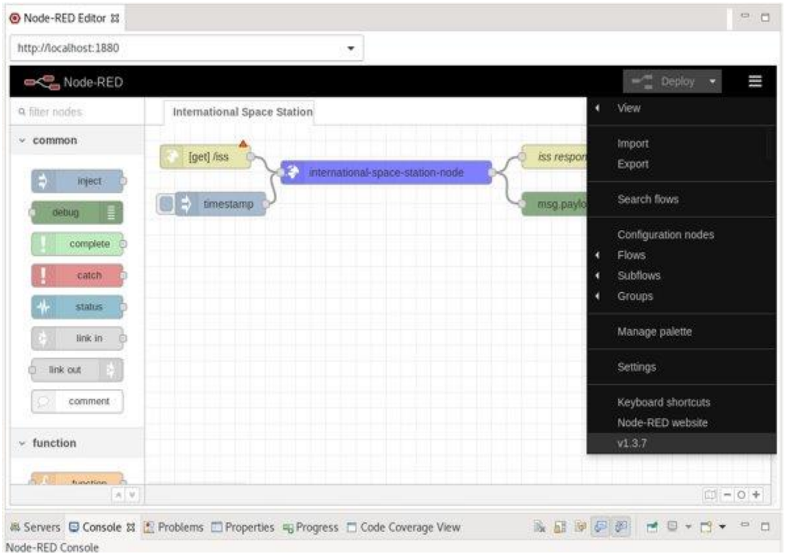
Name	Type	Value
this		{TCF: com.hcl.xtools.transform.tcjs.impl.TCF@1b57c345
targetPlatform		"Win64_MinGW"
allTCs		[Lcom.hcl.xtools.transform.tcjs.impl.JSTC;@21ea996f
topTC		ID =platform:/resource/ExecutableProject1/buildTopE...
Global		

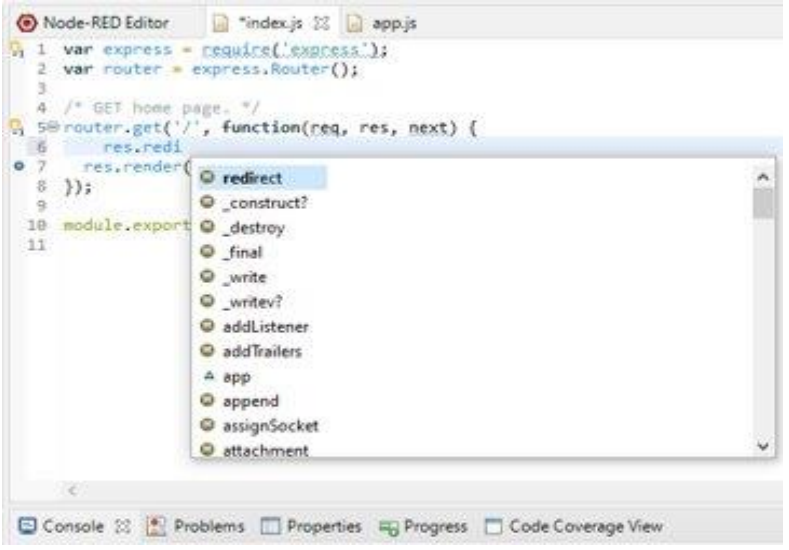
HCL SOFTWARE

NodePlus 2.0

RTist only

- 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





```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
    res.redi
  res.render(
});

module.export
```

HCL SOFTWARE

HCL

Relationship™

BEYOND THE CONTRACT

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

WATCH THE **FILM**