

# HCLSoftware


## What's New in DevOps Model RealTime 12.1.7

updated for the DevOps 2026.06 release

# Overview

- ▶ Model RealTime 12.1 is based on Eclipse 2024-06 (4.32)
- ▶ Two brandings of the product are available (HCL and IBM). There is no differences in functionality between them.
  - The only difference is in the licensing mechanism and branding (e.g. documentation)



 DevOps Model RealTime  
Version: 12.1.7 (part of DevOps 2026.06)  
Build: 12.1.7.v20260605\_0649

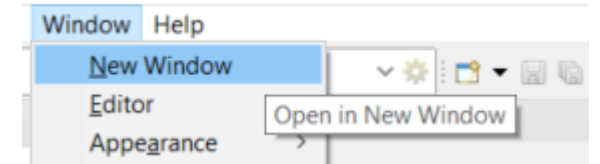
(c) Copyright IBM Corporation 2004, 2016. All rights reserved.  
(c) Copyright HCL Technologies Ltd. 2016, 2026. All rights reserved.  
Visit <https://model-realtime.hcldoc.com/help/topic/com.ibm.xtools.rsarte.webdoc/users-guide/overview.html>

# Eclipse 4.32 (2024.06)

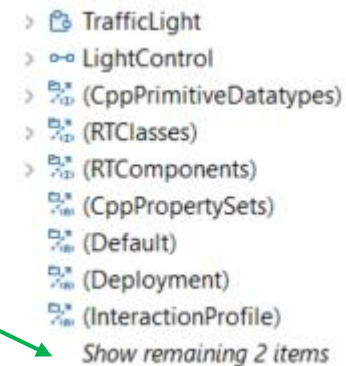
- ▶ Compared to Model RealTime 12.0, version 12.1 includes new features and bug fixes from 4 quarterly Eclipse releases:
  - 2023.09 (<https://www.eclipse.org/eclipse/news/4.29/platform.php>)
  - 2023.12 (<https://www.eclipse.org/eclipse/news/4.30/platform.php>)
  - 2024.03 (<https://www.eclipse.org/eclipse/news/4.31/platform.php>)
  - 2024.06 (<https://www.eclipse.org/eclipse/news/4.32/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 4.32 (2024.06)

- ▶ Better support for opening additional workbench windows when using multiple monitors
  - Now a new workbench window is opened on the same monitor where the current workbench window is located
  - Previously it opened on the primary monitor, which could be "far away" from the current workbench window
- ▶ The number of items shown in many views can now be limited to improve UI performance
  - Controlled by a new preference **General - Initial maximum number of elements shown in views** (by default 1000)
  - Affects for example the Problems and the Project Explorer views
  - Helps avoiding performance problems when a huge number of elements are shown in these views
  - You can disable this feature by setting the preference to 0

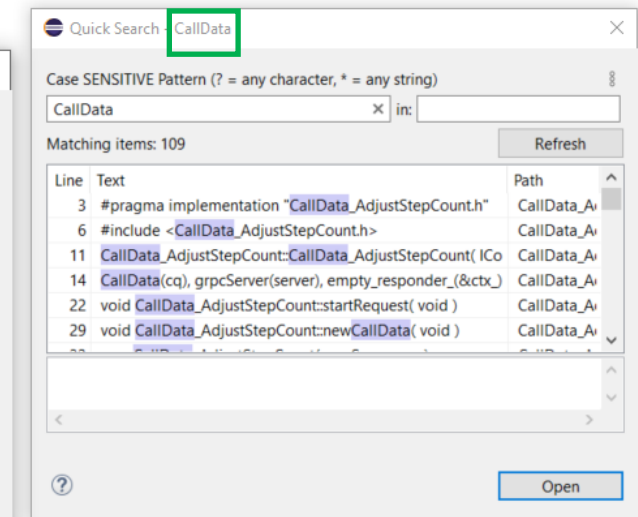
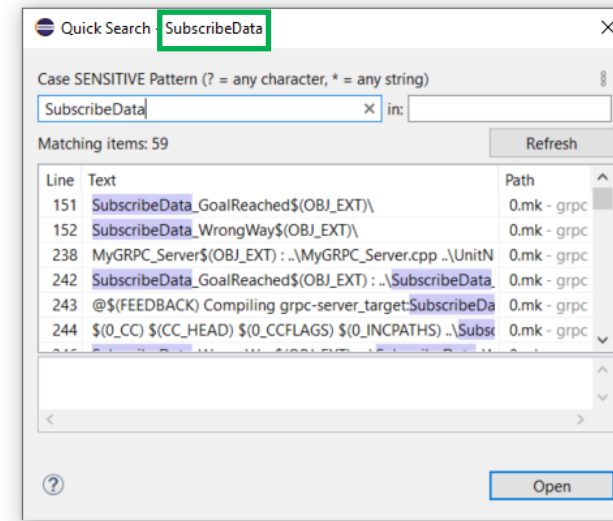
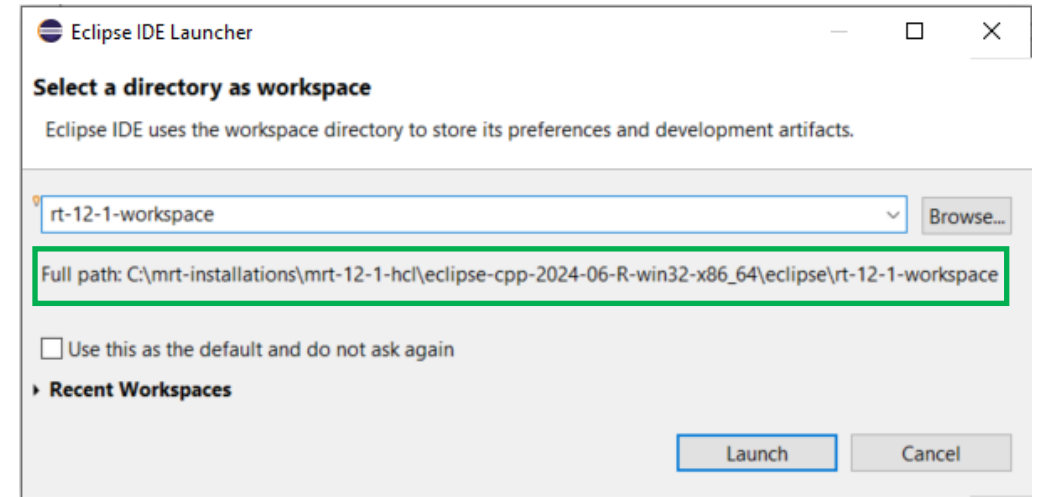


*click here to show more items*



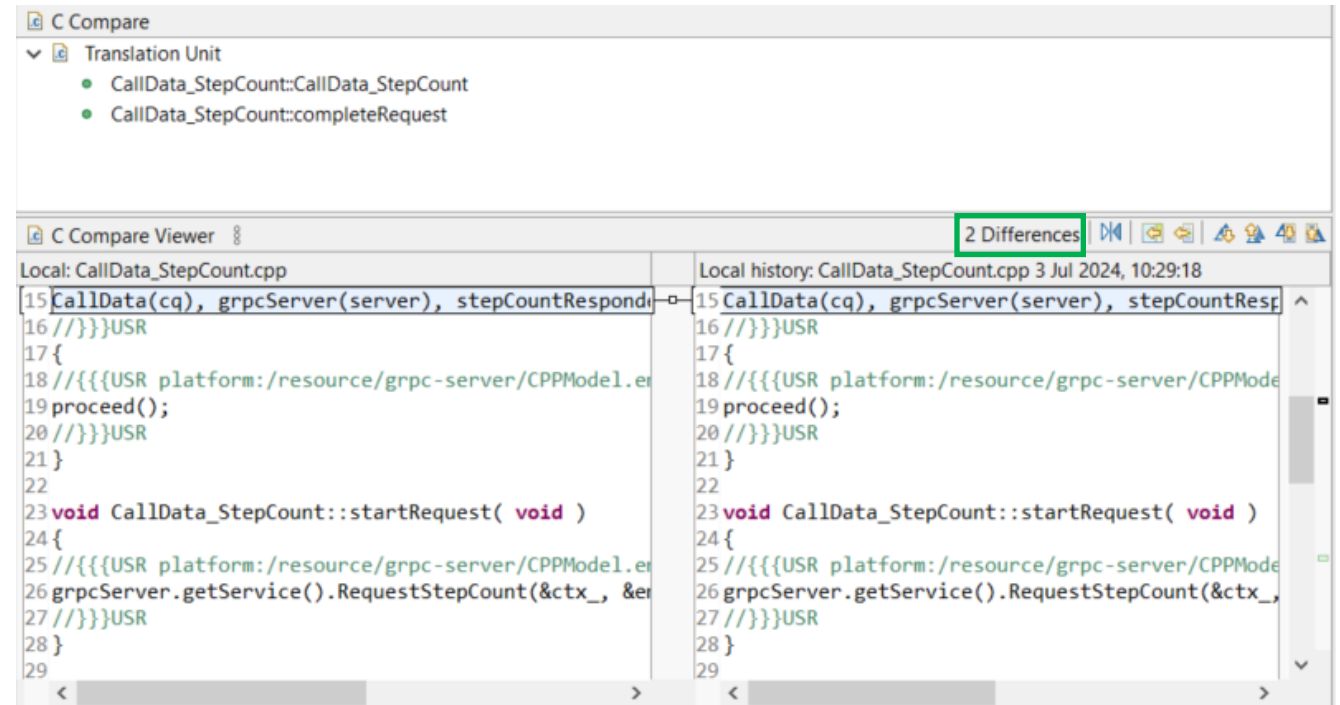
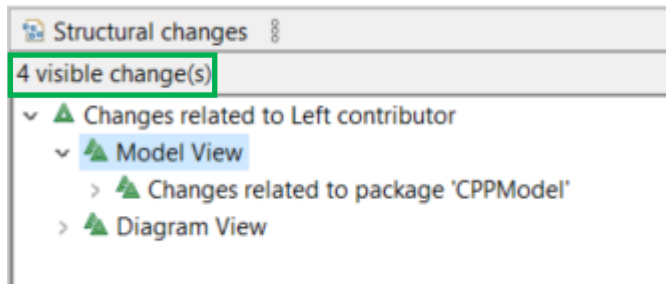
# Eclipse 4.32 (2024.06)

- ▶ The launch dialog now shows the resolved path of the workspace that will be opened
  - Useful if you use relative paths or the ~ character on Linux/MacOs (for specifying the home folder)
  - Also tells you if you happen to use a character in the workspace name that is not valid on your operating system
- ▶ Multiple Quick Search dialogs can now be more easily distinguished since the search term is printed in their titles
  - Useful if you tend to keep a few of those modeless dialogs open for repeated searches

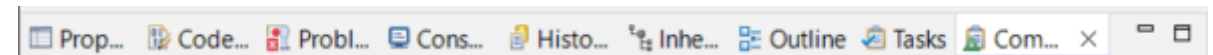


# Eclipse 4.32 (2024.06)

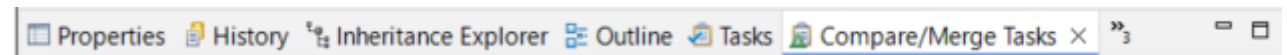
- ▶ The Compare/Merge editor for text files now shows the total number of differences
  - This was already before shown in the Compare/Merge editor for models



- ▶ A new preference can be set to avoid very short titles when a large number of views appear in the same view stack
  - **General - Appearance - Always show full titles**
  - Another new preference allows to hide the view icons to further save space



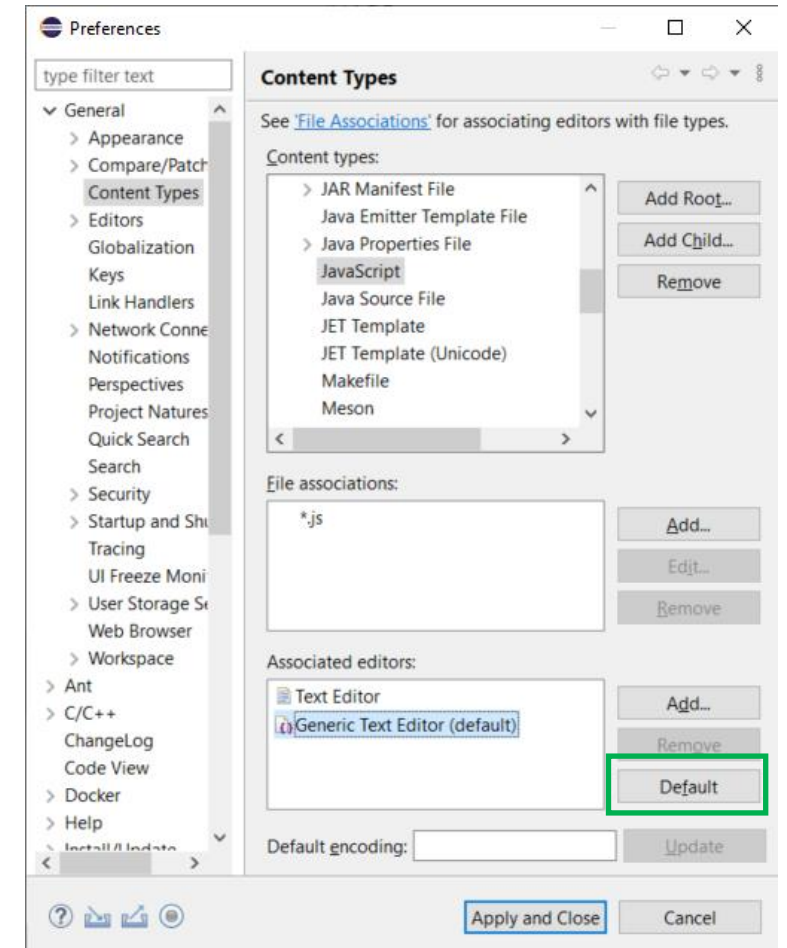
*before (and still the default)*



*now (with new preference set)*

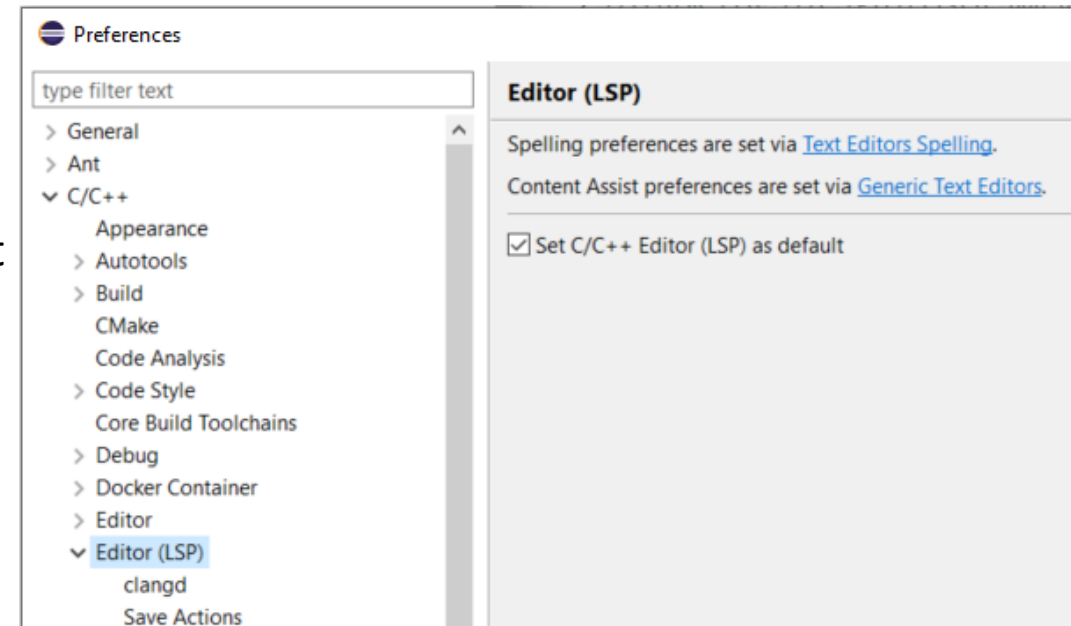
# Eclipse 4.32 (2024.06)

- ▶ It's now possible to change the default editor for a certain file content type
  - For example, if you use Build Variants, but don't have a JavaScript plugin installed, you can now create a content type for the \*.js file extension and set the Generic Text Editor as the default editor
  - Previously it was necessary to always use the **Open With** command which was easy to forget
- ▶ Restart of Eclipse now works the same as an exit followed by a relaunch
  - For example, changes in `eclipse.ini` are now picked up when restarting



# CDT 11.6 (included as part of Eclipse 2024.06)

- ▶ Information about the CDT improvements can be found here:  
<https://github.com/eclipse-cdt/cdt/blob/main/NewAndNoteworthy/CDT-11.3.md>  
<https://github.com/eclipse-cdt/cdt/blob/main/NewAndNoteworthy/CDT-11.4.md>  
<https://github.com/eclipse-cdt/cdt/blob/main/NewAndNoteworthy/CDT-11.5.md>  
<https://github.com/eclipse-cdt/cdt/blob/main/NewAndNoteworthy/CDT-11.6.md>
- ▶ **Note:** Eclipse 2024.06 also contains another plugin for C/C++ development called [CDT-LSP](#) which makes use of the Clang language server (clangd)
  - You have to enable this feature in the Preferences - by default CDT is still used for new projects
  - You can use both CDT and CDT-LSP in the same workspace for different C++ projects
  - Model RealTime does not yet support CDT-LSP (i.e. target projects are still always using CDT, and the Code View/Editor also uses CDT)



# Newer EGit Version in the EGit Integration

- ▶ The EGit integration in Model RealTime has upgraded EGit from 6.6 to 6.10
  - This is the recommended and latest version for Eclipse 2024.06
- ▶ This upgrade provides several new features and bug fixes
  - For detailed information about the changes see
    - [https://wiki.eclipse.org/EGit/New\\_and\\_Noteworthy/6.7](https://wiki.eclipse.org/EGit/New_and_Noteworthy/6.7)
    - [https://wiki.eclipse.org/EGit/New\\_and\\_Noteworthy/6.8](https://wiki.eclipse.org/EGit/New_and_Noteworthy/6.8)
    - <https://projects.eclipse.org/projects/technology.egit/releases/6.9.0>
    - <https://projects.eclipse.org/projects/technology.egit/releases/6.10.0>

# Java 21

---

- ▶ Eclipse 2024.06 requires a JVM for Java 21 or newer
  - It comes with a Java 21 JVM included which will be used by default
  - Model RealTime therefore also requires Java 21 and it's recommended to use that exact version. A warning dialog will appear on start-up in case another JVM version is used.
    - This check can be disabled by setting the system property `com.ibm.xtools.umltdt.core.disableJavaCheck`

# MacOS

---

- ▶ The Model RealTime user interface can be used on MacOS
  - Non-experimental support for both the x86\_64 and Arm architectures
- ▶ All known MacOS specific defects have now been resolved
- ▶ Default values for unspecified TC properties have been added for MacOS. This makes it faster to get started with building applications on MacOS.

# BIRT Reporting

- ▶ The BIRT Integration can now be used with Model RealTime 12.1.x
  - Delivered as a separate update site on the Info Center

## Create, manage, and run configurations

Create a configuration to generate a BIRT report.

The screenshot shows the BIRT configuration window for 'Report RT diagrams'. The 'Report' section has 'Built-In: UML Model Diagram Report' selected. The 'Description' field contains: 'Demonstrates how to include diagrams images from a UML model, re image resolution. This is a strategy that can be used to balance diagram memory.' The 'Report Data' section shows 'Data Sources' with '\*Main model' and 'Instance Models' with 'TrafficLightsDemo/TrafficLight Analysis Model.emx' and 'TrafficLihtsDemo/TrafficLihtComponent.emx'.

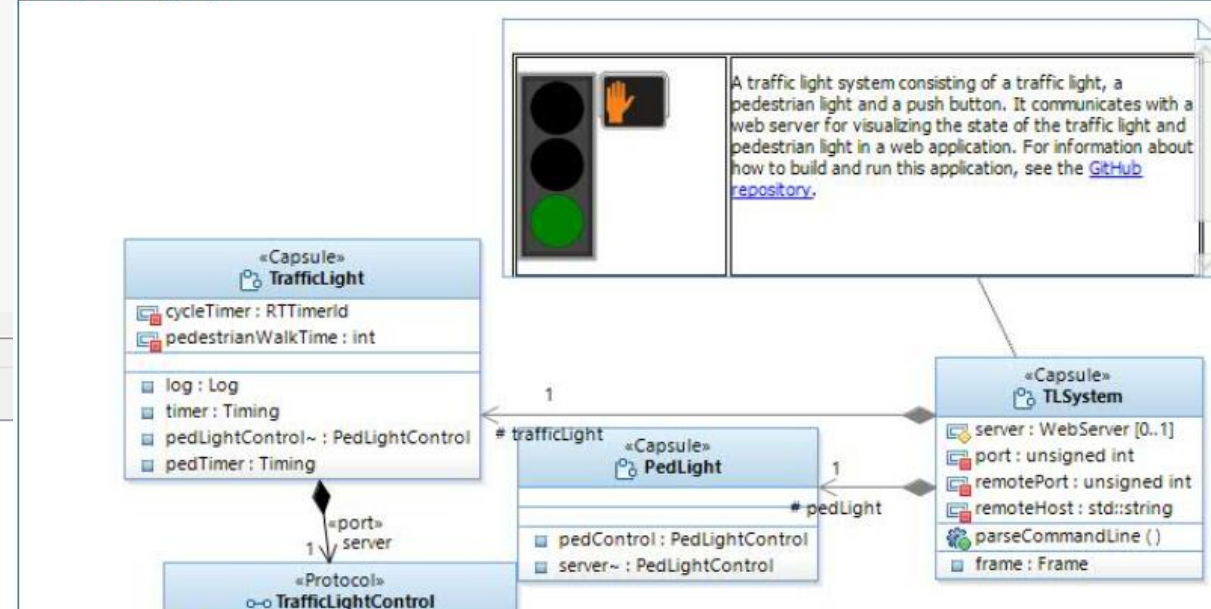
## TrafficLight Analysis Model::Collaboration1::Interaction1

### SequenceDiagram1 (Sequence Diagram)



## TrafficLightComponent

### Main (Class Diagram)



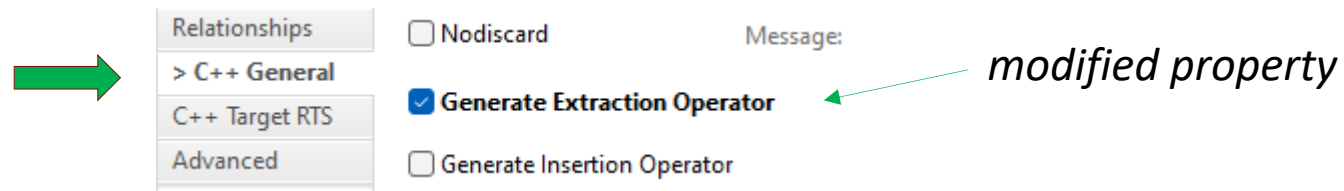
# Removal of the One Test Embedded Integration

---

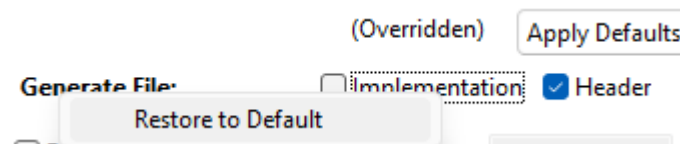
- ▶ The One Test Embedded Integration does not work with Eclipse 2024.06 and has therefore been removed from Model RealTime
  - The [documentation on the Info Center](#) will remain for some time for users still using One Test Embedded together with older versions of Model RealTime

# Visual Indication of Modified Properties in the Properties View

- ▶ The property tabs “C++ General” and “C++ Target RTS” contain a large number of properties
  - They affect how code is generated and are therefore more “important” than some of the properties elsewhere
- ▶ These tab names are now prefixed with “>” if any of these properties contain a non-default value
  - Makes it quick and easy to detect elements with custom property values
  - Avoid having to click and scroll these property tabs to find out

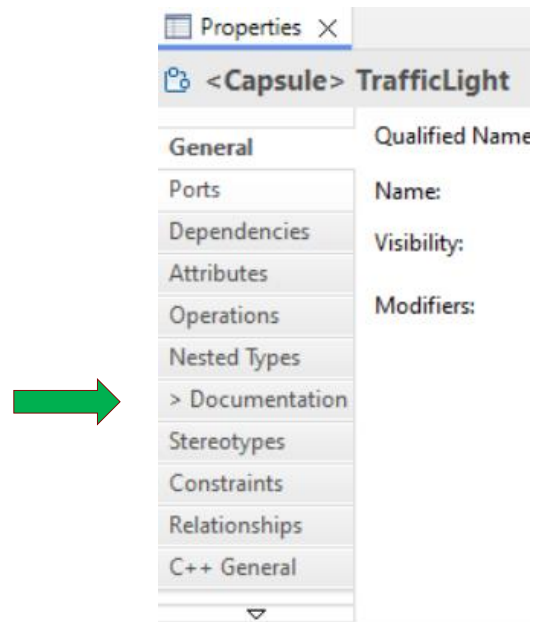


- ▶ Also, it’s now possible to restore a modified property to its default value by means of a new context menu command **Restore to Default**:



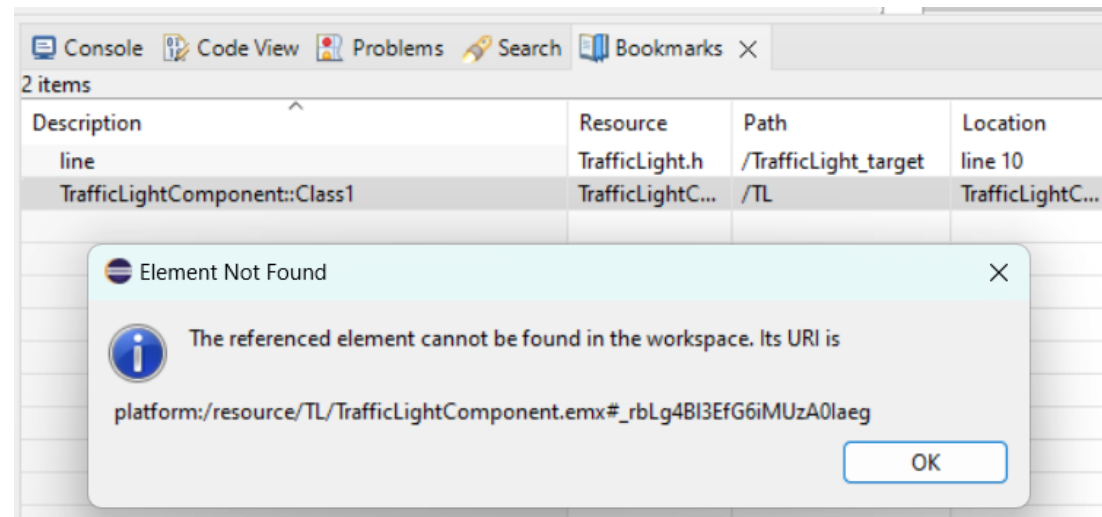
# Visual Indication of the Presence of Documentation in the Properties View

- ▶ The property tab “Documentation” now shows a “>” prefix if a documentation comment is present for a model element
  - Makes it easier to find where documentation is available and avoids unnecessary clicks on the Documentation tab



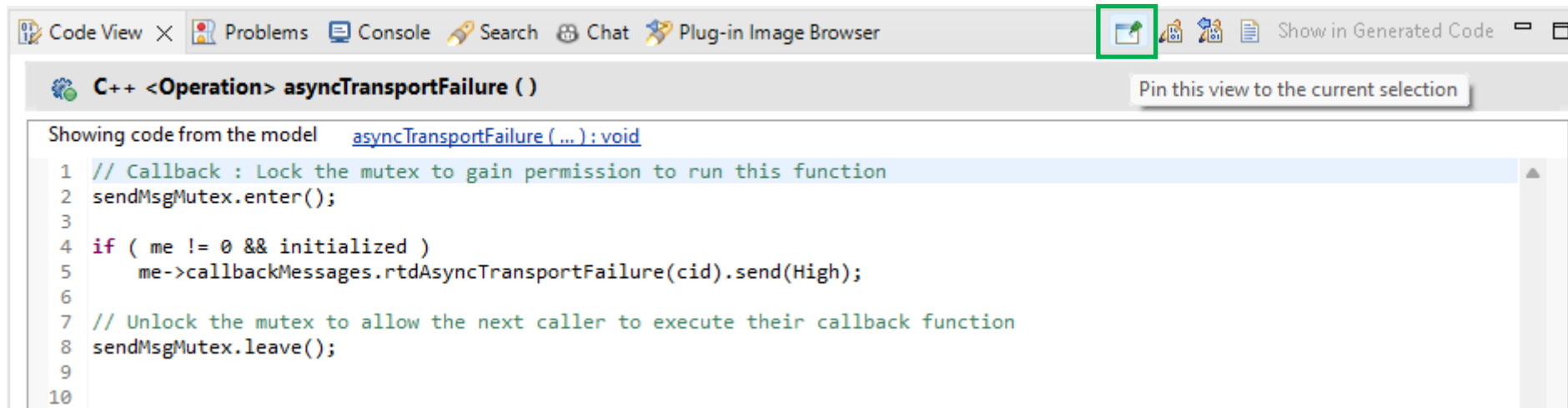
# Bookmark Improvements

- ▶ The Bookmarks view is now updated to remove bookmarks for elements that are no longer present in the workspace
  - For example, elements that have been deleted or whose projects are currently closed
- ▶ If anyway navigation for a bookmark fails, a dialog now appears and shows the URI of the missing element



# Pin Code View to Selection

- ▶ The “Pin to selection” button is now available in the Code view toolbar
  - Works the same as for the Properties view, i.e. locks the Code view to always show code for a certain element
- ▶ Pinning the Code view can for example be useful if you want to navigate to a called function to check its implementation, but then return to working at the code snippet that calls the function
  - The navigation changes the selection, and previously it was necessary to manually find the element in the Project Explorer to select it again



The screenshot shows the Eclipse IDE interface. The top toolbar includes buttons for Code View, Problems, Console, Search, Chat, and Plug-in Image Browser. A green box highlights the 'Pin this view to the current selection' button, which is represented by a pin icon. Below the toolbar, the Code View is open for the C++ operation `asyncTransportFailure()`. The code snippet is as follows:

```
Showing code from the model asyncTransportFailure \(...\): void  
1 // Callback : Lock the mutex to gain permission to run this function  
2 sendMsgMutex.enter();  
3  
4 if ( me != 0 && initialized )  
5     me->callbackMessages.rtdAsyncTransportFailure(cid).send(High);  
6  
7 // Unlock the mutex to allow the next caller to execute their callback function  
8 sendMsgMutex.leave();  
9  
10
```

# TC Editor Improvements (1/2)

- ▶ The Sources table now has columns to more easily manage the list of source elements
  - Possible to sort the source elements by clicking the column headers
  - The Qualified Name and Resource columns help distinguish multiple source elements with the same name

Sources

Qualified Name	Name	Resource
 CPPModel::Client	Client	platform:/resource/rtBound/CppModel.emx#_Won7...
 CPPModel::PROT::PROT	PROT	platform:/resource/rtBound/CppModel.emx#_xLyT...
 CPPModel::Server	Server	platform:/resource/rtBound/CppModel.emx#_vN8tk...
 CPPModel::Top	Top	platform:/resource/rtBound/CppModel.emx#_VJr4c...

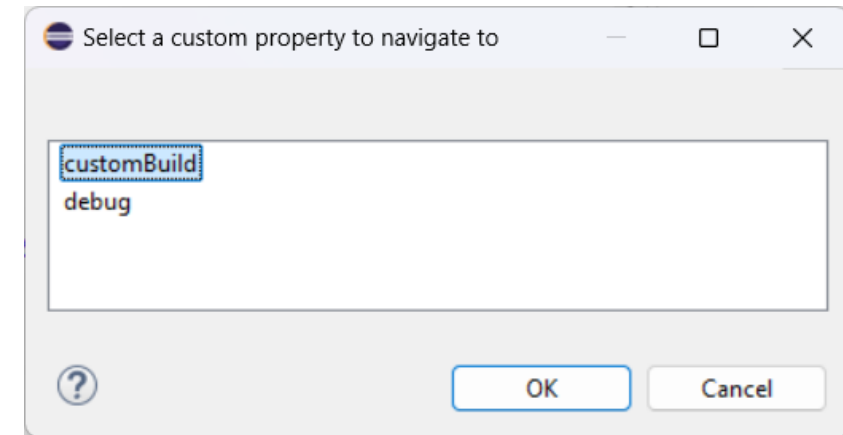
# TC Editor Improvements (2/2)

- ▶ The presence of custom TC properties is now easier to detect thanks to a new hyperlink
  - It appears at the top of the TC editor if the TC has one or many custom properties
  - Click the hyperlink to navigate to the custom property (or properties)
  - Custom properties are for example useful when using Build Variants

[C++ Executable] app.tcjs

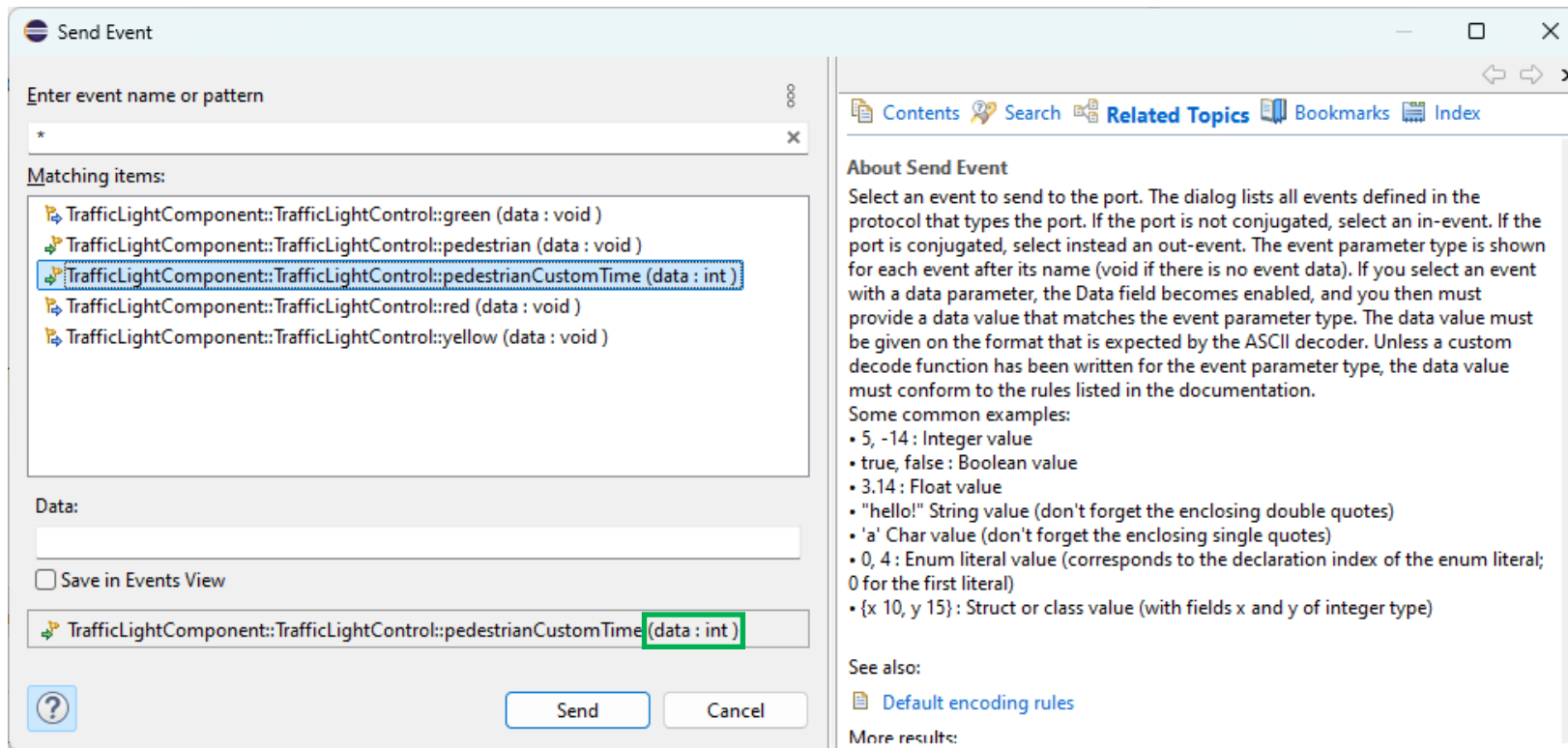
[More custom properties present in Code tab](#)

```
1 let tc = TCF.define(TCF.CPP_TRANSFORM);
2 tc.sources = [
3   'platform:/resource/TL/TrafficLightComponent.emx#_ayAoEg2EfCGm_vSAbO-SQ',
4 ];
5 tc.createTargetProject = true;
6 tc.customBuild = true;
7 tc.debug = true;
8 tc.targetProject = '/TrafficLight_target';
9 tc.topCapsule = 'platform:/resource/TL/TrafficLightComponent.emx#_aynxkg2EfCGm_vSAbO-SQ';
```



# Improved Send Event Dialog

- ▶ The Send Event dialog now shows the parameter type for each listed event
  - Makes it easier to know what to type in the Data field when sending an event
- ▶ Context Sensitive Help is now also available for this dialog



The screenshot displays the 'Send Event' dialog box. On the left, there is a search field 'Enter event name or pattern' containing an asterisk. Below it, a list of 'Matching items' is shown, each with a small icon and a parameter type in parentheses: 'TrafficLightComponent::TrafficLightControl::green (data : void)', 'TrafficLightComponent::TrafficLightControl::pedestrian (data : void)', 'TrafficLightComponent::TrafficLightControl::pedestrianCustomTime (data : int)', 'TrafficLightComponent::TrafficLightControl::red (data : void)', and 'TrafficLightComponent::TrafficLightControl::yellow (data : void)'. The 'pedestrianCustomTime' item is highlighted with a blue border. Below the list is a 'Data:' field, a checkbox for 'Save in Events View', and a preview of the selected item: 'TrafficLightComponent::TrafficLightControl::pedestrianCustomTime (data : int)'. At the bottom are 'Send' and 'Cancel' buttons, and a help icon. On the right, a help pane titled 'About Send Event' provides instructions on selecting events and formatting data values. It lists examples: integer values (5, -14), boolean values (true, false), float values (3.14), string values ('hello!'), char values ('a'), enum literals (0, 4), and struct/class values ({x 10, y 15}). A 'See also' link for 'Default encoding rules' is provided.

# Model Compiler Console Improvements

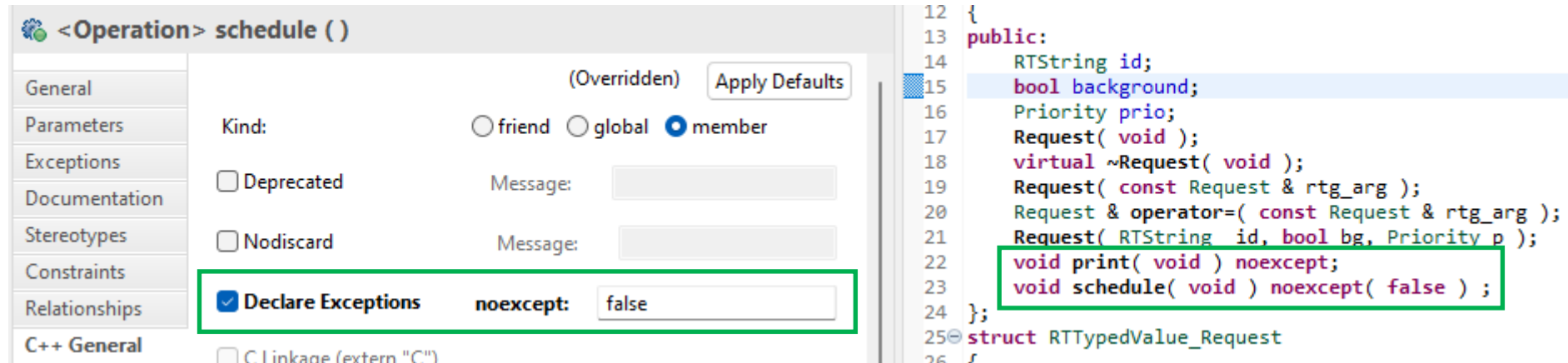
- ▶ Errors and warnings are no longer printed to the Model Compiler console
  - They are now only printed in the UML Development console
- ▶ Avoids printing too much information in that console which could make the interesting information on top "scroll away" to become non-accessible
  - Note that the Model Compiler console cannot be increased in size (so therefore must be kept short), while the UML Development console can be increased in size with **Preferences - Run/Debug - Console - Limit console output**

```
Model Compiler
16:48:36 : INFO : New MCTask 4 : Process app.tcjs
16:48:36 : INFO : 16:48:36: Exporting diagram layouts...
16:48:36 : INFO : Processing {/TL/app.tcjs} with model compiler server.
16:48:36 : INFO : Use this command line to perform the same task with stand-alone Model Compiler:
C:/model-rt-installations/mrt_hcl_12_1_6/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0...
16:48:36 : INFO : Output log file:
C:\eclipse-workspaces\wmf_export\.metadata\.plugins\com.ibm.xtools.umldt.ui\mc.messages\B07AB32F1BF1CC96033C931A6F430...
16:48:36 : INFO : 16:48:36: Diagram layout export completed successfully.
16:48:37 : INFO : MCTask 4 completed successfully with Exit code = 0, byte[38] details contain [messages]
```

*This is the most important information shown in the Model Compiler console, and it now remains visible even if a huge number of errors/warnings are reported*

# Generating Functions with “noexcept” Specifiers

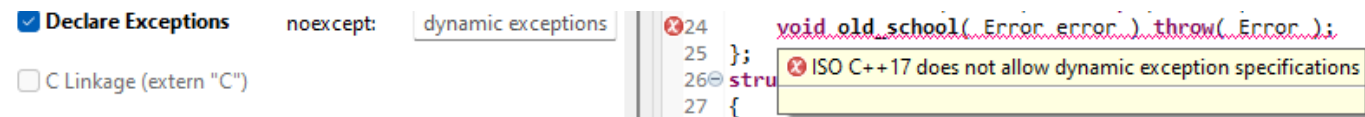
- ▶ It's now possible to specify that an operation cannot throw any exceptions, or may throw exceptions, by means of the noexcept specifier



```
12 {
13 public:
14     RTString id;
15     bool background;
16     Priority prio;
17     Request( void );
18     virtual ~Request( void );
19     Request( const Request & rtg_arg );
20     Request & operator=( const Request & rtg_arg );
21     Request( RTString id, bool bg, Priority p );
22     void print( void ) noexcept;
23     void schedule( void ) noexcept( false );
24 };
25 struct RTTypedValue_Request
26 }
```

- ▶ For backwards compatibility, dynamic exception specifications are still supported

- But this feature was removed in C++ 17, and the Model Compiler will therefore print a warning if you attempt to use it with a too new code standard



```
24 void old_school( Error error ) throw( Error );
25 };
26 struct RTTypedValue_Request
27 {
```

WARNING : ISO C++17 does not allow dynamic exception specifications

WARNING : Dynamic exception specifications should not be used with C++ 17 or later. Consider using the noexcept specifier instead.

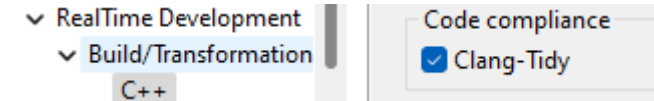
- ▶ Use of noexcept requires C++ 11 or later

- The Model Compiler will print a warning if you attempt to use it with a too old code standard

WARNING : Use of the noexcept specifier requires C++ 11 or later.

# Code Compliance

- ▶ Two additional Clang-Tidy rules are now supported when the preference **RealTime Development – Build/Transformations – C++ - Clang-Tidy** is set



- **misc-use-anonymous-namespace**

Suppress warnings for use of the `static` keyword for non-member functions in generated .cpp files

```
static void _rtg_deleteThreads( void ) /* NOLINT(misc-use-anonymous-namespace) */
```

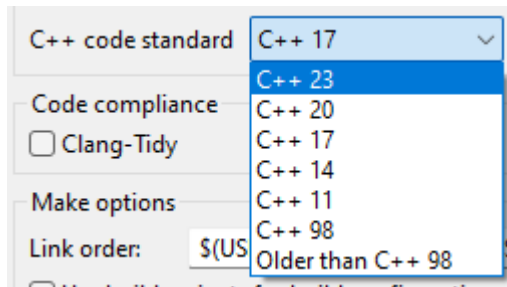
- **bugprone-branch-clone**

Suppress warnings related to switch-statements in generated .cpp files

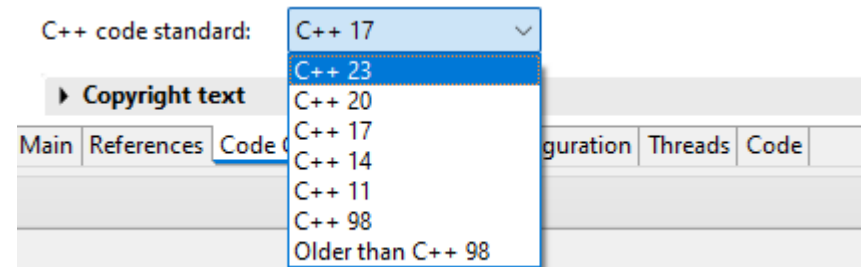
```
// NOLINTBEGIN(bugprone-branch-clone);  
for (int stateIndex=getCurrentState(); ;stateIndex=rtg_parent_state[stateIndex - 1]){  
    switch( stateIndex )
```

# C++ 23

- ▶ C++ 23 is now available as a new code standard to use
  - For both the TC setting and the workspace preference
  - Currently no features in Model RealTime require this new code standard, but if generated code is compiled with C++ 23, it's useful to be able to set this as the code standard to be used for code generation too (for consistency)



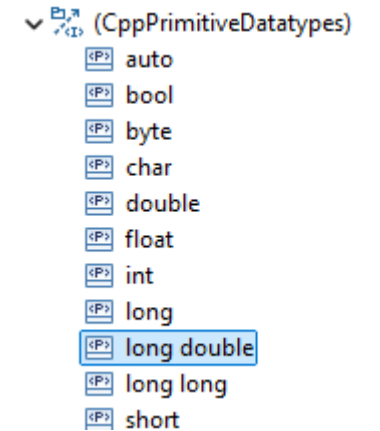
*workspace preference*



*TC setting*

# Long Double

- ▶ The TargetRTS and Model Compiler now supports the primitive type `long double`
  - It was already available to be used in the `CppPredefined` package, but was previously ignored by the code generator (and not supported by the TargetRTS)
  - The Encoding, Decoding and Logging APIs were extended to support `long double` in the same way as other primitive types are supported



# Log Streams

- ▶ The TargetRTS now provides two log streams `Log::out` and `Log::err`
- ▶ They provide a number of benefits compared to logging using traditional log ports
  - Available in all code, not just in capsule code snippets where log ports can be accessed
  - Consistent way of logging either to `std::out` or `std::err`
  - Easier to create log messages that contain a mix of text and data
  - Based on `std::ostream` – all standard C++ stream manipulators can be used
  - Possible to lock/unlock log streams to avoid interleaved log messages in multi-threaded applications
  - Possible to print log messages to a log file by means of log streams

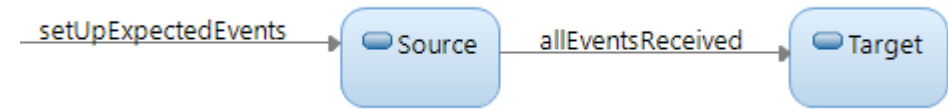
```
Log::out << "Status: " << std::boolalpha << result.getStatus() << Log::endl;  
Log::err << Log::lock << "Running in thread " << context()->name() << Log::endl << Log::unlock;  
MyType mt(4);  
Log::out << "MyType value: " << RTTypedValue(&mt, &RTType_MyType) << std::endl;
```

# Changed Encoding of Some Primitive Types

- ▶ Primitive types that have a space in their name (long long, long double etc) are now encoded so that the space is replaced with an underscore (\_)
  - For example, the value 4 of type long long was previously encoded as “4 long long” but is now instead encoded as “4 long\_long”
- ▶ This change was done to ensure that encoded strings only contain one space that separates the value from the type name
  - Previously it was not possible to send events in the Model Debugger where the event parameter had a primitive type with a space in its name, but with this change it’s now working correctly
- ▶ Note: If your application somehow relies on the old encoding for these types it must be updated to accommodate for this change!

# Waiting for Multiple Events

- ▶ A new utility class in the TargetRTS, `RTMultiReceive`, makes it easy to transition from a source state to a target state only when a certain set of expected events have been received
  - Previously it was usually necessary to store information in capsule variables and use choices in the state machine for implementing this pattern
- ▶ Set-up the expectations on which events to receive when entering the source state
  - Use the class `RTEventReception` to express a static constraint on whether a received event is expected or not
  - Use the class `RTEventReceptionMsg` to express a dynamic constraint where a predicate function can decide if a received event is expected or not based on the received `RTMessage`



# Change the Application Exit Code

---

- ▶ Previously the exit code of a generated real-time application was always 0
- ▶ Now the TargetRTS class `RTMain` provides an API for getting and setting the application exit code
  - `RTMain::getExitCode()`
  - `RTMain::setExitCode()`
- ▶ The default exit code is still 0

# Custom Priority for Event Raised on External Port

- ▶ Previously when code in an external thread raised an event on an external port, the message would always get the default priority (`General`)
- ▶ Now the capsule that owns the external port can set the priority by a new optional parameter for the `enable()` function

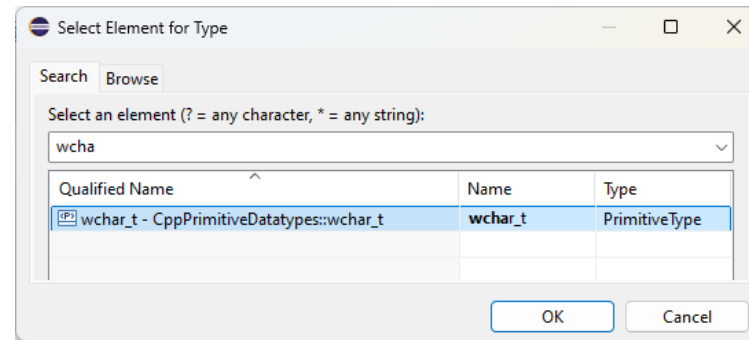
```
void enable(int prio = System);
```

- ▶ This makes it possible to handle messages from external threads with a custom priority

# Support for wide characters

- ▶ The TargetRTS now supports use of wide characters, and arrays of wide characters, using the primitive type `wchar_t`.
- ▶ A type descriptor for this type is available so it for example can be encoded and decoded

DataClass  
msg: wchar\_t [3] { L'飼', L'館', L'鶴' }



RT C++ Target Run Time System - Release 8.0.14

Json encoding: {"msg" : ["飼", "館", "鶴"]}

# Port full warning

- ▶ A new configuration setting for the TargetRTS is available: **PORTFULL\_WARNING**
- ▶ If set, a runtime warning will be printed if you attempt to connect unwired ports in ways that violate their multiplicities.
  - For example, if you have an SPP port with multiplicity 1 that two SAP ports try to connect to, then this warning is reported to inform that the second SAP port will remain unbound.

```
RT C++ Target Run Time System - Release 8.0.16
```

```
targetRTS: observability listening not enabled
```

```
Task 0 detached
```

```
WARNING: SAP cap[1]:Cap.sap cannot connect to SPP application[0]:Top.spp with size 1 at this time (no free port index on SPP).
```

# Registration of custom debug error function

- ▶ The RTMain class now provides a function for registering a "debug" error function:  
`registerDebugErrorFunction()`
- ▶ A registered function will be called for all error messages printed to `stderr` in `RTDebugger::trace()`
  - The feature can be used for capturing such error messages and report them in some custom way (for example writing them to a file)

# RTS Debugger Documentation

- ▶ The Run Time System Debugger (RTS Debugger) is included by default in every executable that is built with Model RealTime
  - It's a very old feature, but was previously not covered in the documentation
- ▶ Now the RTS Debugger is documented here: <https://secure-dev-ops.github.io/code-realtime/running-and-debugging/rts-debugger>

```
>.\Top.exe
```

```
RT C++ Target Run Time System - Release 8.0.12
```

```
targetRTS: observability listening not enabled
```

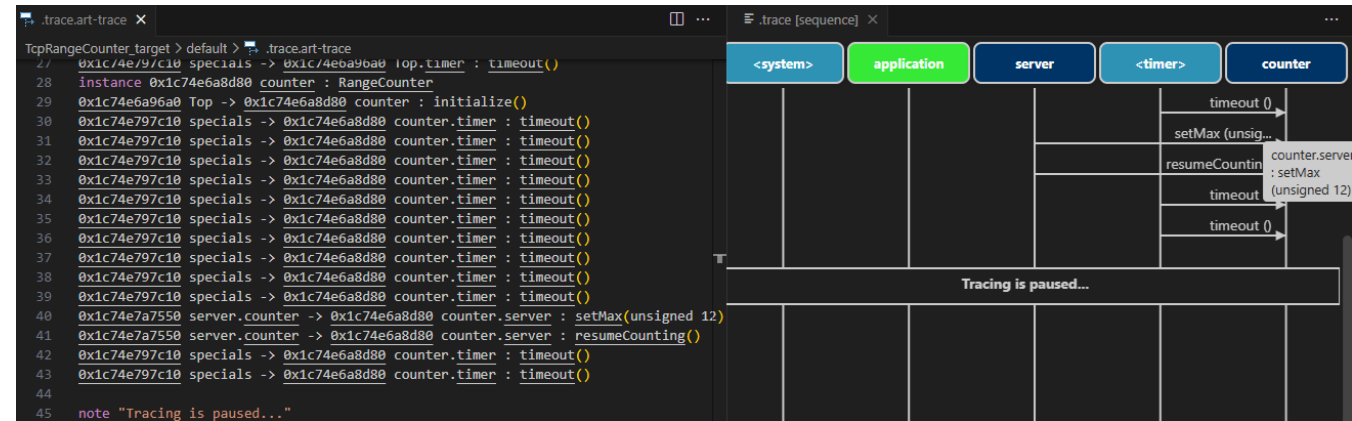
```
RTS debug: ->
```

# Tracing and Sequence Diagram Visualization (1/3)

- ▶ A new trace feature, focusing on the message exchange between capsule instances at run-time
  - Implemented in the TargetRTS and is independent of the Model Debugger UI
  - Captured traces can be visualized graphically as sequence diagrams
  - Useful for troubleshooting bugs, investigation of performance problems, dead-locks, detection of dead code etc.
- ▶ A trace file format **.art-trace** is supported
  - Can be opened in Code RealTime and visualized there as a sequence diagram
  - Control the frequency of flushing the trace file by means of a new TargetRTS configuration setting: [RTTRACER\\_FLUSH\\_COUNT](#)
- ▶ This tracing can be turned on/off from
  - Command-line
  - RTS Debugger
  - Code snippets (programmatically)
  - Model Debugger
  - C++ Debugger

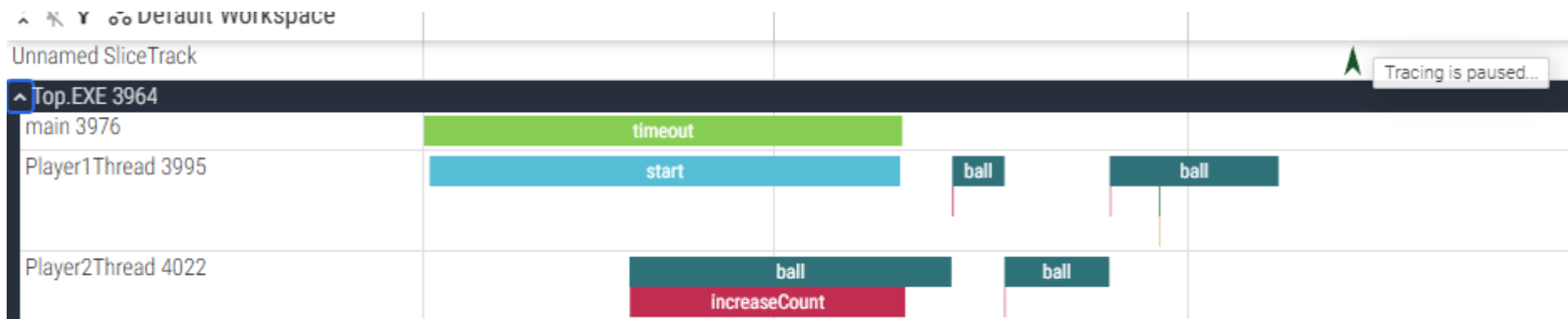
# Tracing and Sequence Diagram Visualization (2/3)

- ▶ Code RealTime sequence diagram viewer for .art-trace files
  - Can handle big trace files (lazy load of “pages”)
  - Tooltips with detailed information
  - Double-click navigation
  - Support for showing both asynchronous and synchronous communication
  - Filter instances and messages using regular expressions
- ▶ A trace configuration file can be provided to customize what information to capture in the trace
  - Use a new command-line argument `-traceConfig=<trace config file>`
  - Timestamps for when messages are received and handled can be turned on (useful for measuring the time it takes for a capsule instance to handle a received message)



# Tracing and Sequence Diagram Visualization (3/3)

- ▶ Captured trace files can be exported to the Google Trace Event format
  - A popular trace format supported by many trace viewers, for example Perfetto



- ▶ For more information about the tracing feature, see <https://secure-dev-ops.github.io/code-realtime/running-and-debugging/tracing>

# Test Runner Script

- ▶ A Node.js script is now available for automated testing of
  - Custom TargetRTS changes (making sure your changes to the TargetRTS work correctly)
  - Code RealTime libraries you may use in Model RealTime (traditional regression testing)
- ▶ The script can either be used in headless mode (useful when run from CI/CD tools) or in an interactive mode through a web browser (useful during development)
- ▶ See <https://github.com/secure-dev-ops/code-realtime/tree/main/art-comp-test/runner> for the script and documentation on how to use it

## mc-executable

Testing running for 2 minutes 29 seconds

Terminate Server Run All Run All Failed Clean All

Hide Passed

**Executive Summary:** Total: 166 Passed: 104 Failed: 1

### abstract\_capsule

Capsules can be declared as abstract by means of the `abstract` keyword. Such a capsule cannot be incarnated, but you can use it as a base for other capsules. Certain validation rules that check the correctness of a capsule state machine are not run for abstract capsules, since those are expected to be specialized in derived capsules.

Completed (passed)	<a href="#">Build Log</a>	<a href="#">Test Log (stdout)</a>	<a href="#">Test Log (stderr)</a>	17 sec	Run	Clean
--------------------	---------------------------	-----------------------------------	-----------------------------------	--------	-----	-------

### abstract\_capsule\_method\_signatures

Checking signatures of pure virtual functions for analysis of abstract while extending.

Completed (passed)	<a href="#">Build Log</a>	<a href="#">Test Log (stdout)</a>	<a href="#">Test Log (stderr)</a>	17 sec	Run	Clean
--------------------	---------------------------	-----------------------------------	-----------------------------------	--------	-----	-------

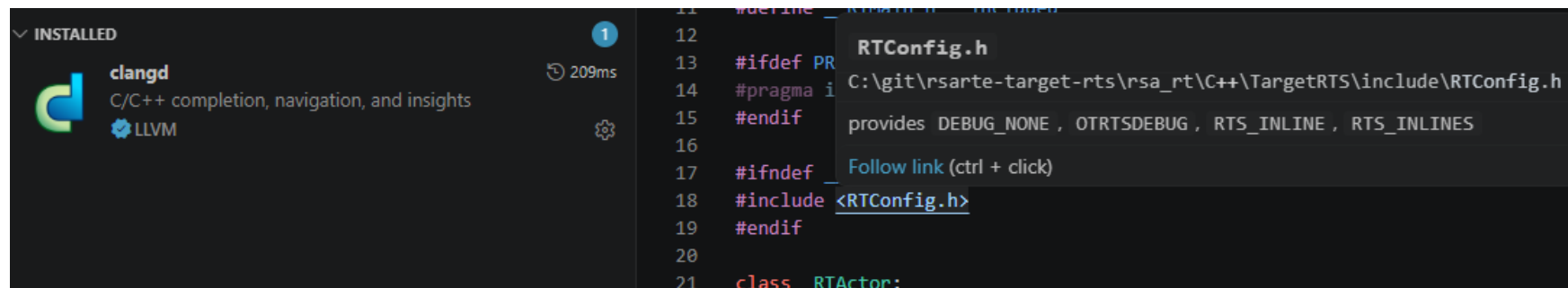
### art\_0001

Test validation rule ART\_0001\_invalidNameCpp.

Completed (passed)	<a href="#">Build Log</a>			2 sec	Run	Clean
--------------------	---------------------------	--	--	-------	-----	-------

# View and Edit TargetRTS Source Files in Visual Studio Code

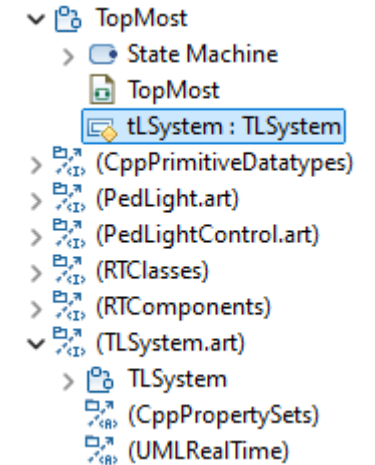
- ▶ The TargetRTS now provides a script (Node JS): `generate-compile-commands.js`
  - Takes a target configuration as input and generates a compilation database for the Clangd language server (`compile_commands.json`)
  - For example: `node generate-compile-commands.js --target WinT.x64-MinGw-12.2.0`
- ▶ After running the script you can open the TargetRTS as a workspace folder in Visual Studio Code (with the Clangd extension installed) and directly work with TargetRTS source files
  - The compilation database enables Clangd to resolve all references and all editing features to work correctly
- ▶ See also <https://github.com/secure-dev-ops/code-realtime/target-rt/#working-with-the-source-files>



The screenshot shows the Visual Studio Code interface. On the left, the 'EXTENSIONS' view is open, showing the 'clangd' extension installed. The extension description reads 'C/C++ completion, navigation, and insights' and 'LLVM'. On the right, a code editor displays C++ preprocessor directives. A tooltip is visible over the `#include <RTConfig.h>` line, showing the path `C:\git\rsarte-target-rt\rsa_rt\C++\TargetRTS\include\RTConfig.h` and a 'Follow link (ctrl + click)' option. The code in the editor includes lines for `#define`, `#ifdef`, `#pragma`, `#endif`, and `class RTActor;`.

# Support for Art Files (Code RealTime Integration)

- ▶ A larger subset of the Art language is now supported
  - Use Art protocols (as type of UML-RT ports, or to inherit UML-RT protocols from)
  - Art events with user-defined parameter types can now be used
  - Use Art capsules as type of UML-RT parts. Connect ports of the Art capsule to UML-RT ports.
- ▶ The Art Compiler is now automatically invoked from generated make file
  - No longer necessary to manually build in Code RealTime if Art or TC files are changed
  - Requires Model RealTime to know where Code RealTime is installed (by means of a new string substitution variable ART\_COMPILER)

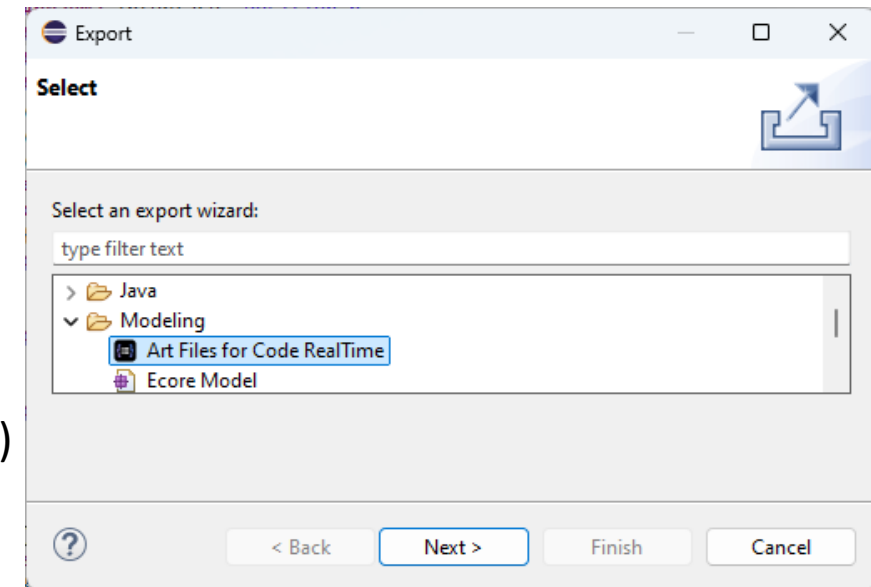


String Substitution			
Create and configure string substitution variables.			
Variable	Value	Description	Contributed By
ART_COMPILER	C:\codert\build\vscode-codert-202-ms\data\extensions\s...	Absolute path to the art compiler jar file 'artcompiler.jar'	com.ibm.xtool...
PCA_RT_HOME[read-only]	C:\model-it-installations\12.1.1-241017\collins\res...		com.ibm.xtool...

- ▶ Note: This feature replaces the experimental feature “Support for textual state machines” which now has been removed

# Art Exporter

- ▶ A new utility for exporting your models to Art files for Code RealTime
- ▶ Available as a separate update site to be installed on top of Model RealTime
  - Get it from the [Utilities page on the Info Center](#)
- ▶ Examples of situations when it can be useful
  - Exporting data types from a model to reuse them in a Code RealTime application or library
  - Migrate a Model RealTime application to Code RealTime (most model constructs, diagram layouts and TC properties are now supported)
- ▶ The Art Exporter can either be run from the Eclipse UI as an export wizard, or from the command-line by calling the model compiler with a new `--exportArt` option
  - When run from the UI it's possible to export diagram layouts for state machine diagrams



# HCLSoftware

[hcltechsw.com](https://www.hcltechsw.com)