

# What's New in HCL RTist 11.3

updated for release 2023.27

# Overview

- ▶ RTist 11.3 is based on Eclipse 2022.06 (4.24)
- ▶ HCL RTist is 100% compatible with IBM RSARTE and all features in these two products are equivalent.



HCL RTist

Version: 11.3.0.v20230707\_1409

Release: 2023.27

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

(c) Copyright HCL Technologies Ltd. 2016, 2023. All rights reserved.

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

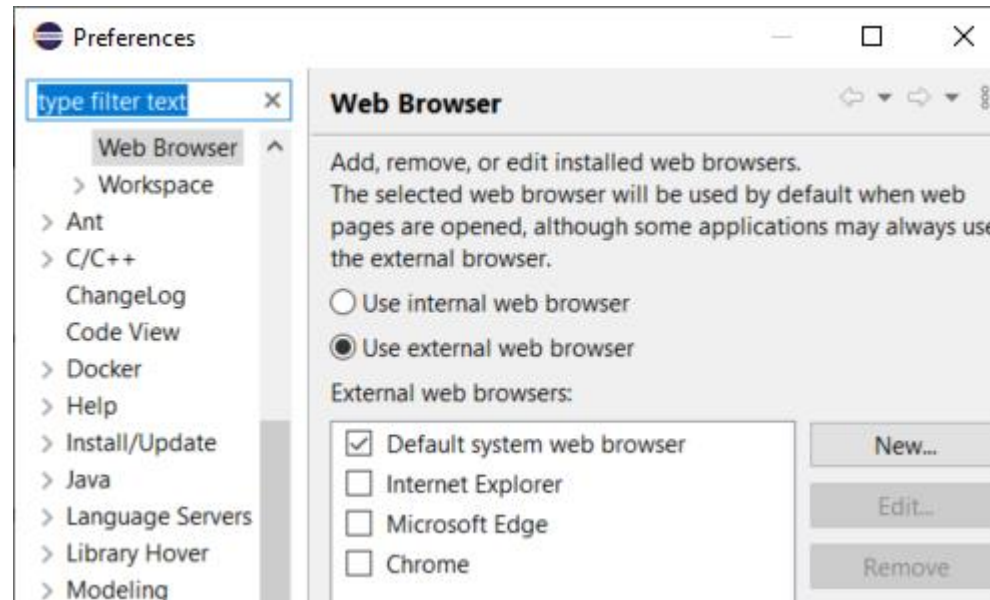
# Eclipse 4.24 (2022.06)

---

- ▶ Compared to RTist 11.2, RTist 11.3 includes new features and bug fixes from 4 quarterly Eclipse releases:
  - 2021.09 (<https://www.eclipse.org/eclipse/news/4.21/platform.php>)
  - 2021.12 (<https://www.eclipse.org/eclipse/news/4.22/platform.php>)
  - 2022.03 (<https://www.eclipse.org/eclipse/news/4.23/platform.php>)
  - 2022.06 (<https://www.eclipse.org/eclipse/news/4.24/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...

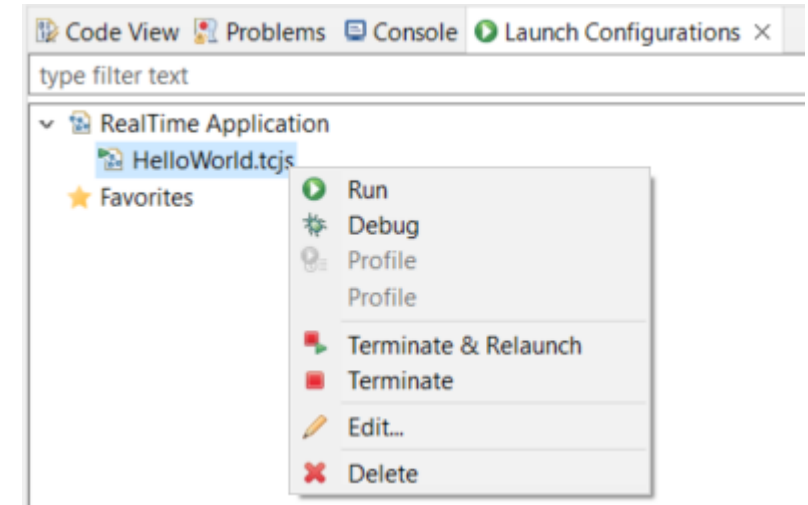
# Eclipse 4.24 (2022.06)

- ▶ Eclipse now by default uses the external web browser
  - The internal web browser has limitations and cannot show all web pages correctly
  - Usage of the external web browser is therefore recommended, and having it as the default makes it easier to get started with RTist without having to configure the preferences in **General - Web Browser**.



# Eclipse 4.24 (2022.06)

- ▶ A new Launch Configuration View makes it easier to, for example, launch model or C++ debug sessions
  - No need to first open the modal Launch Configuration dialog
  - Launch configurations will appear in the view automatically as they are created
  - Commonly useful commands for launching, terminating etc are available in the context menu
  - To start a model debug session, just double-click a "TC launch configuration" in that view



# Eclipse 4.24 (2022.06)

## ► Multiple text selections

- You can now have multiple cursors in a text editor and make multiple selections
- Add a new cursor by **Alt+click**
- Multiple cursors can be useful when making the same change in multiple places in a file ("interactive find/replace")
- Several new text editor commands are available for working with multiple text selections
  - For example: **Multi selection up/down relative to anchor selection** (for creating a multi-selection from a selected word)
  - Note: You need to assign a key binding to these commands to use them! Use the preference page **General - Keys** and search for commands with "multi" in their name in the "Text Editing" category.

```
Showing code from the file Initial
48 log.log("Getting indices");
49 long int i1 = getIndex1();
50 long int i2 = getIndex2();
51 long int i3 = getIndex3();
52 context()->abort();
53
```

```
Showing code from the file Initial
48 log.log("Getting indices");
49 int i1 = getIndex1();
50 int i2 = getIndex2();
51 int i3 = getIndex3();
52 context()->abort();
--
```

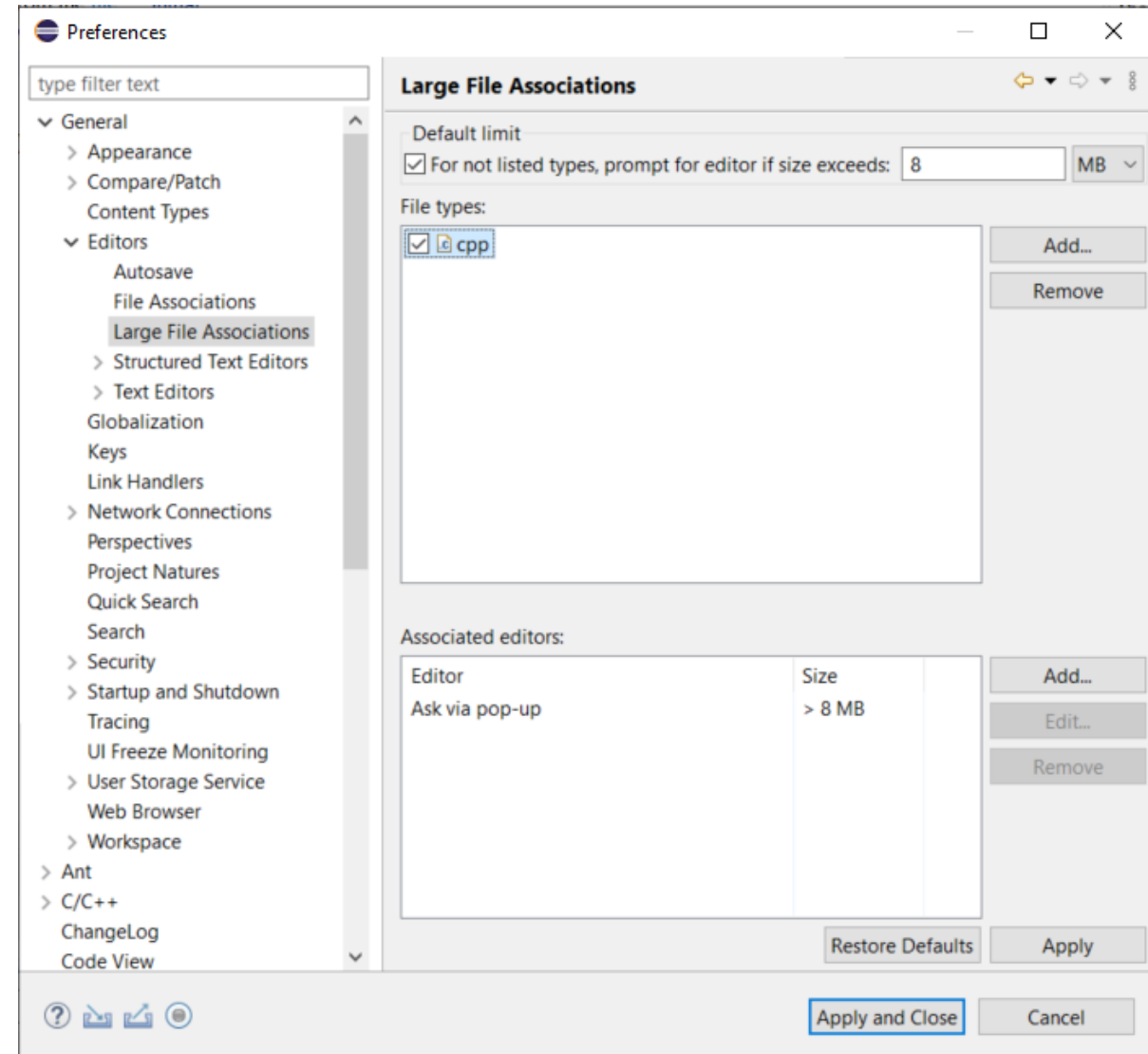
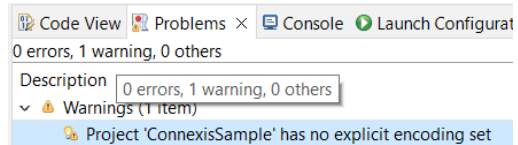
# Eclipse 4.24 (2022.06)

## ▶ Large file associations

- A new preference page allows to specify special editors to use for large files: **General - Editors - Large File Associations**
- Can help keeping a good performance in Eclipse even when opening large files

## ▶ Project encodings

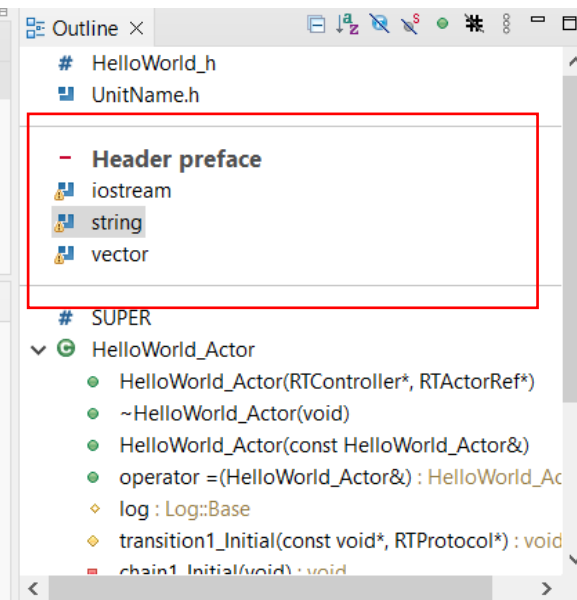
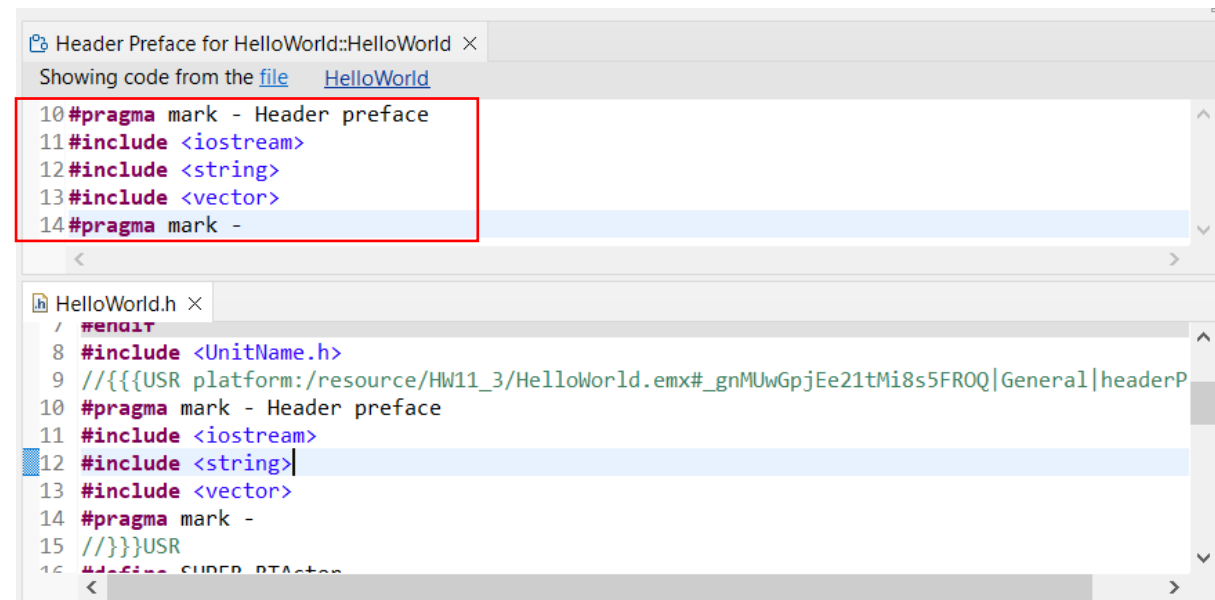
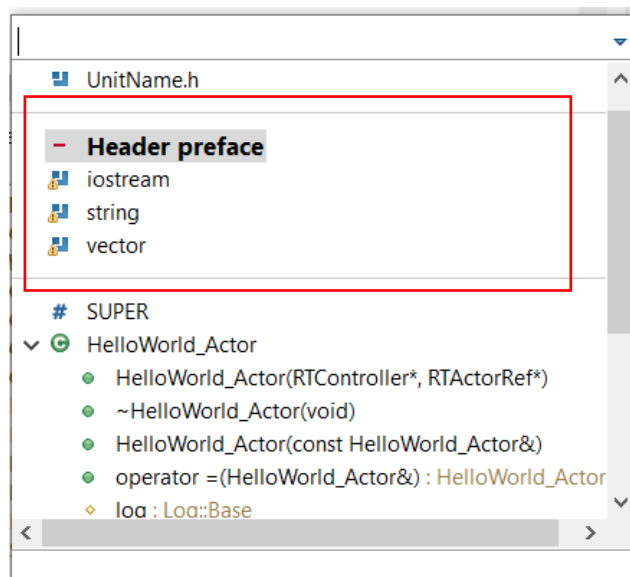
- Projects will now automatically get its encoding set to the workspace encoding (by default UTF-8) when they are created
- For projects created in earlier versions of Eclipse a warning will appear
- A Quick Fix is available for setting the project encoding to the workspace encoding



# CDT 10.7 (included as part of Eclipse 2022.06)

## ▶ Separator lines in the Outline view

- `#pragma mark` and `#pragma region` can be used for showing separator lines in the Outline view
- Can help to more easily see and navigate to user code snippets in a generated C++ file
- Automatically generating such separator lines for certain code snippets could be a future possibility...





# CDT 10.7 (included as part of Eclipse 2022.06)

---

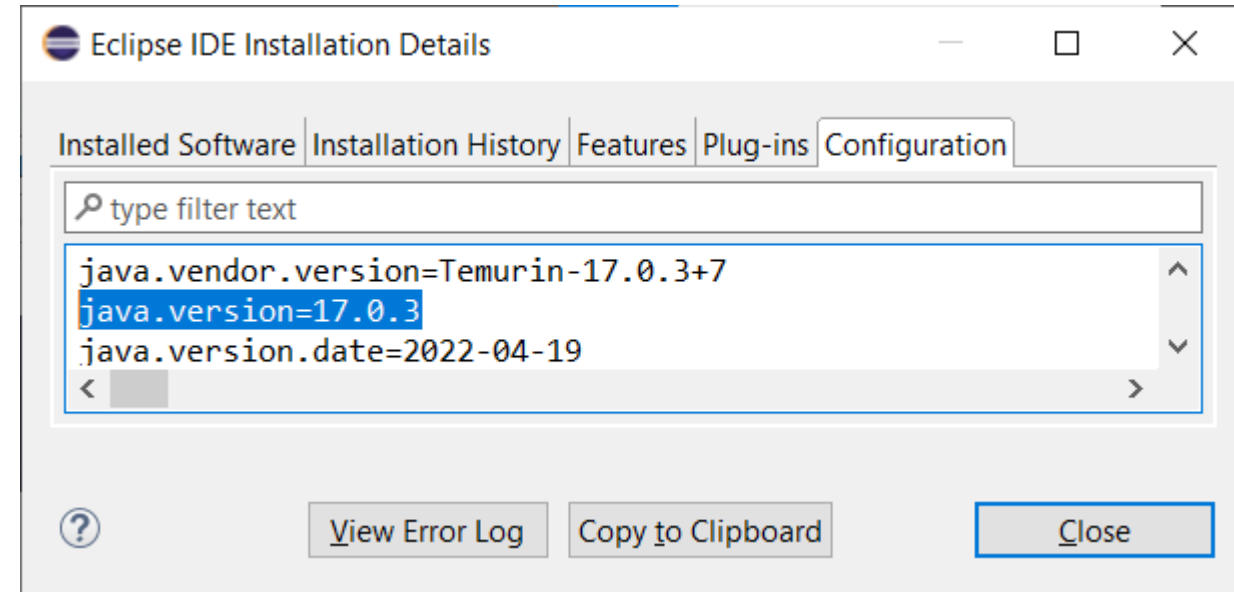
- ▶ Improved code analysis for constexpr expressions
  - A number of GCC/Clang built-in functions can now be used without confusing the Code Analysis feature
- ▶ For more information about CDT improvements see
  - <https://wiki.eclipse.org/CDT/User/NewIn104>
  - <https://wiki.eclipse.org/CDT/User/NewIn105>
  - <https://wiki.eclipse.org/CDT/User/NewIn106>
  - <https://wiki.eclipse.org/CDT/User/NewIn107>

# Newer EGit Version in the EGit Integration

- ▶ The EGit integration in RTist has upgraded EGit from 5.12 to 6.2
  - This is the recommended and latest version for Eclipse 2022.06
- ▶ This upgrade provides several new features and bug fixes
  - For detailed information about the changes see
    - [https://wiki.eclipse.org/EGit/New\\_and\\_Noteworthy/5.13](https://wiki.eclipse.org/EGit/New_and_Noteworthy/5.13)
    - [https://wiki.eclipse.org/EGit/New\\_and\\_Noteworthy/6.0](https://wiki.eclipse.org/EGit/New_and_Noteworthy/6.0)
    - [https://wiki.eclipse.org/EGit/New\\_and\\_Noteworthy/6.1](https://wiki.eclipse.org/EGit/New_and_Noteworthy/6.1)
    - [https://wiki.eclipse.org/EGit/New\\_and\\_Noteworthy/6.2](https://wiki.eclipse.org/EGit/New_and_Noteworthy/6.2)

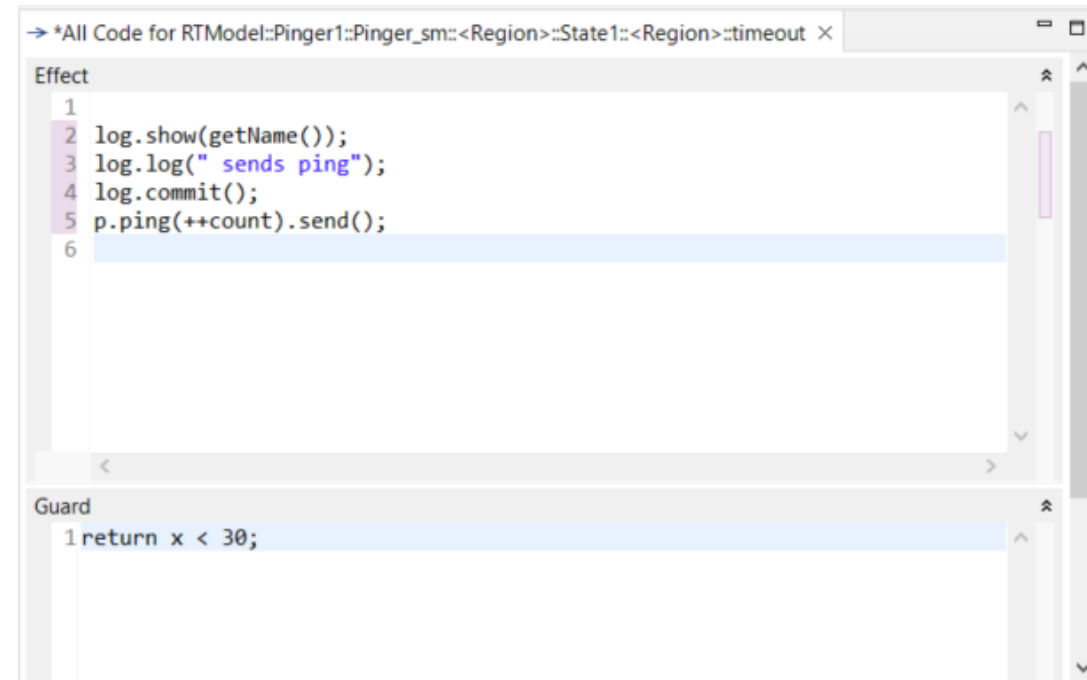
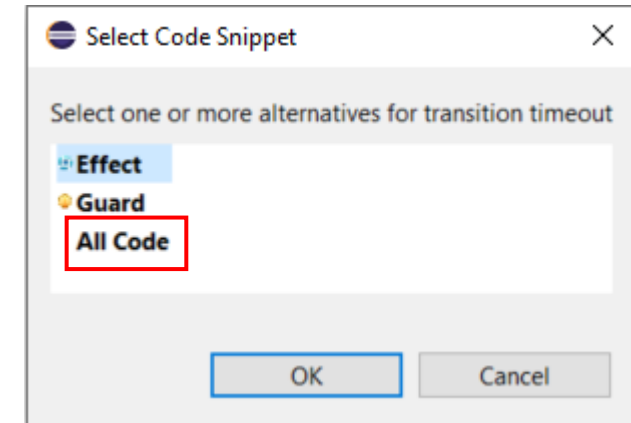
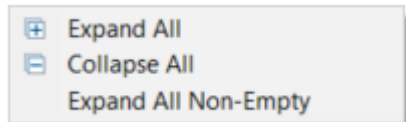
# Java 17

- ▶ RTist now should be run with a Java 17 JVM
  - Eclipse 2022.06 includes a Java 17 JVM which can be used. It's hence no longer necessary to update `eclipse.ini` to specify a different JVM for running RTist.
  - Refer to the System Requirements for more details
- ▶ Rebuild your plugins
  - If you have your own Eclipse plugins it's recommended to rebuild them with a Java 17 compiler before running them together with RTist 11.3.



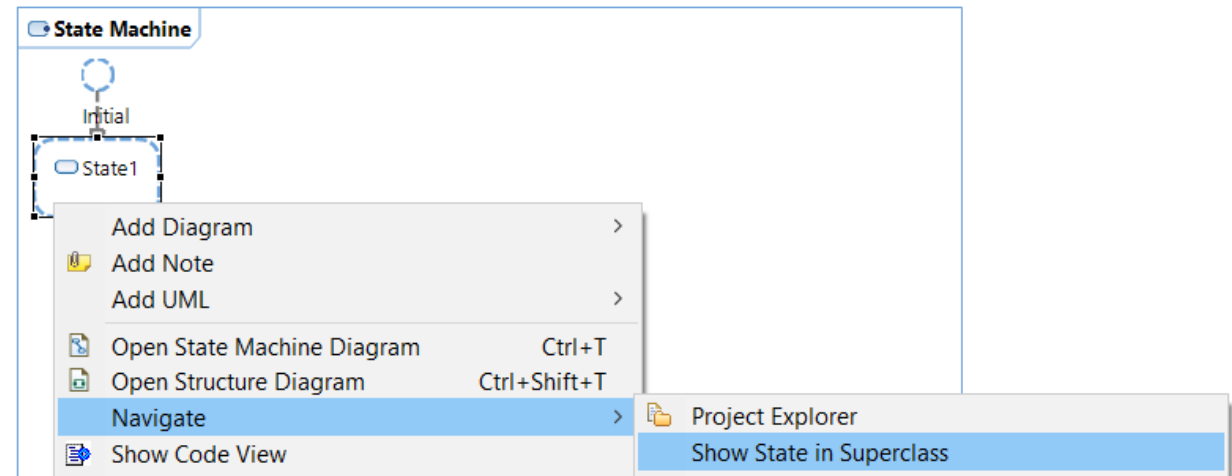
# All Code Editor

- ▶ You can now choose to show all code snippets for an element in a single Code editor
  - Significantly reduces the number of open code editors (e.g. a class has 14 different code snippets which now can be shown in a single code editor, instead of in 14 separate code editors)
  - Better overview by seeing related code snippets in a single place
  - Easier to copy/paste code between related code snippets
- ▶ Individual code snippets can be collapsed and expanded
  - By default empty code snippets are initially collapsed
  - The context menu on a code snippet heading (grey area) contains a few useful commands for collapsing/expanding code snippets:
- ▶ This feature is currently experimental!



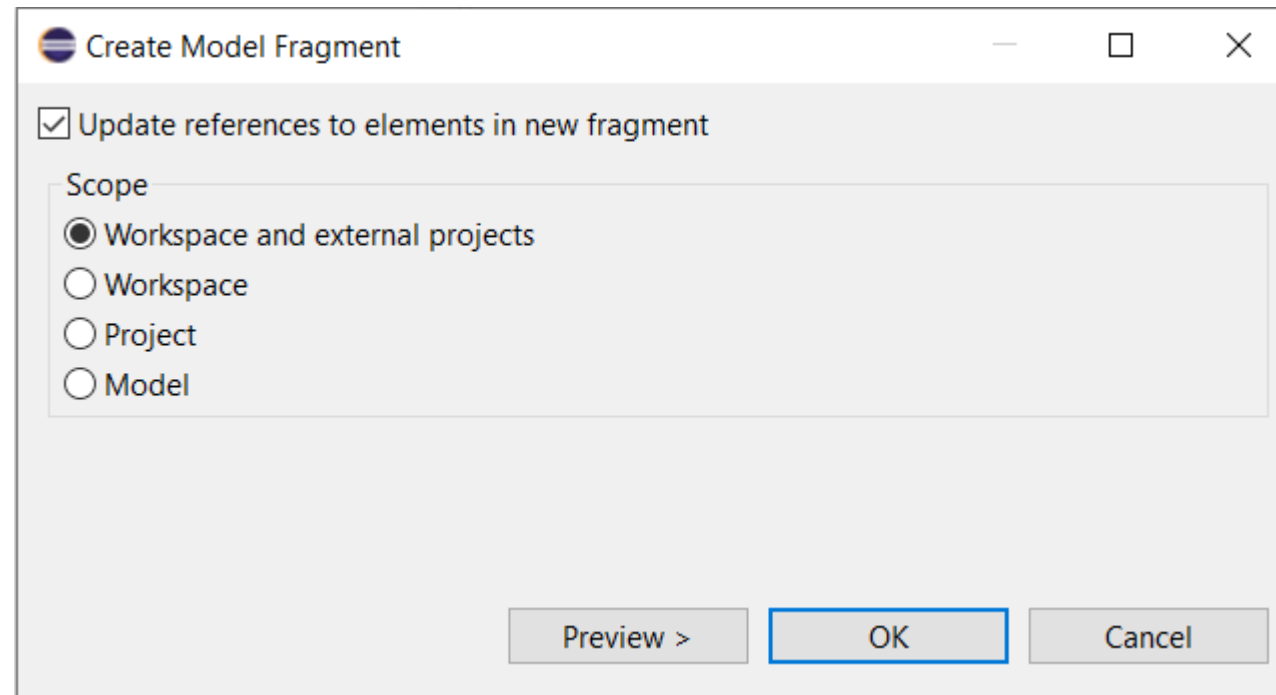
# Navigation to Inherited Elements

- ▶ The context menu command "Show in Superclass" was previously only capable of navigating from a redefining element to the redefined element in the super capsule
- ▶ Now it can also be used for navigating to inherited elements
  - Works for states, transitions and ports
  - Useful for navigating inheritance hierarchies
- ▶ The command is available in both the Project Explorer and diagram context menus
  - The inherited element will be highlighted in the Project Explorer, and from there you can use the context menu command **Navigate - Navigate to Diagram** to view it in a diagram.



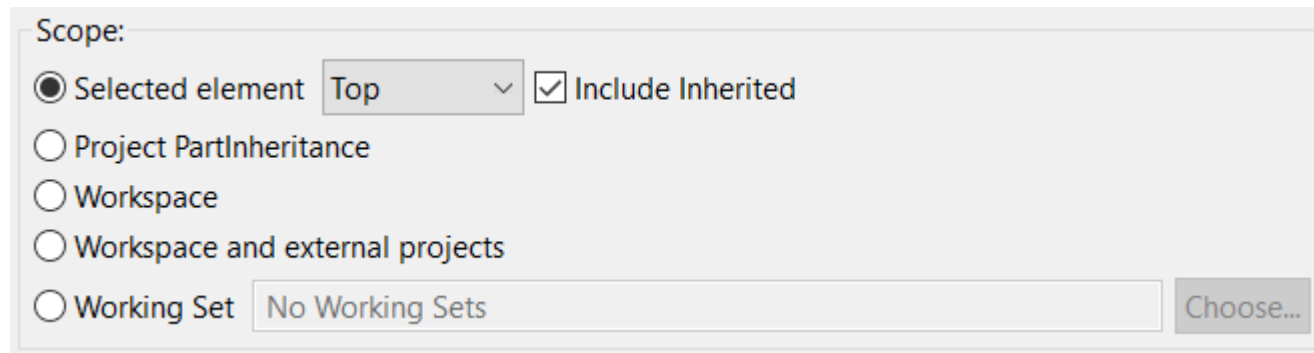
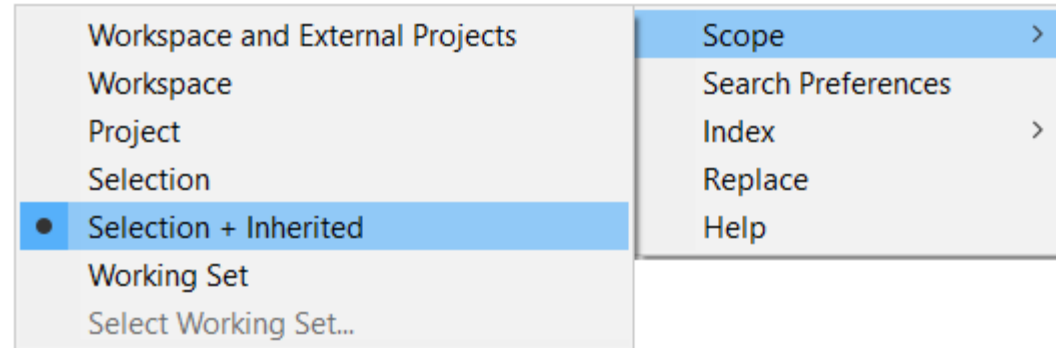
# Improved Refactoring with External Projects

- ▶ Previously certain refactoring commands (e.g. creation of fragments) did not update external projects
  - This has now been solved



# Search in Inheritance Hierarchies

- ▶ It's now possible to search in the scope of an element plus all elements it inherits from
- ▶ For the Search field a new scope "Selection + Inherited" can be selected
- ▶ For the Search dialog a new checkbox "Include Inherited" can be checked
- ▶ This feature works for all elements that support inheritance: capsules, classes and interfaces



# Tracing Port Instances

- ▶ The Model Debugger now supports tracing events that are sent or received on specific port instances
  - Previously only the port itself could be added in the Capture tab in the Trace Editor, but now you can add one or several specific port instances

Capture from selected elements  Capture from all elements

Select the elements to capture trace events from.

Drag/drop elements from the Debug view or use the Add button.

Capture	Element	Path
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> control:0	/ [Top : 0x1d941e2e000]/trafficLight:0/control:0

Filter

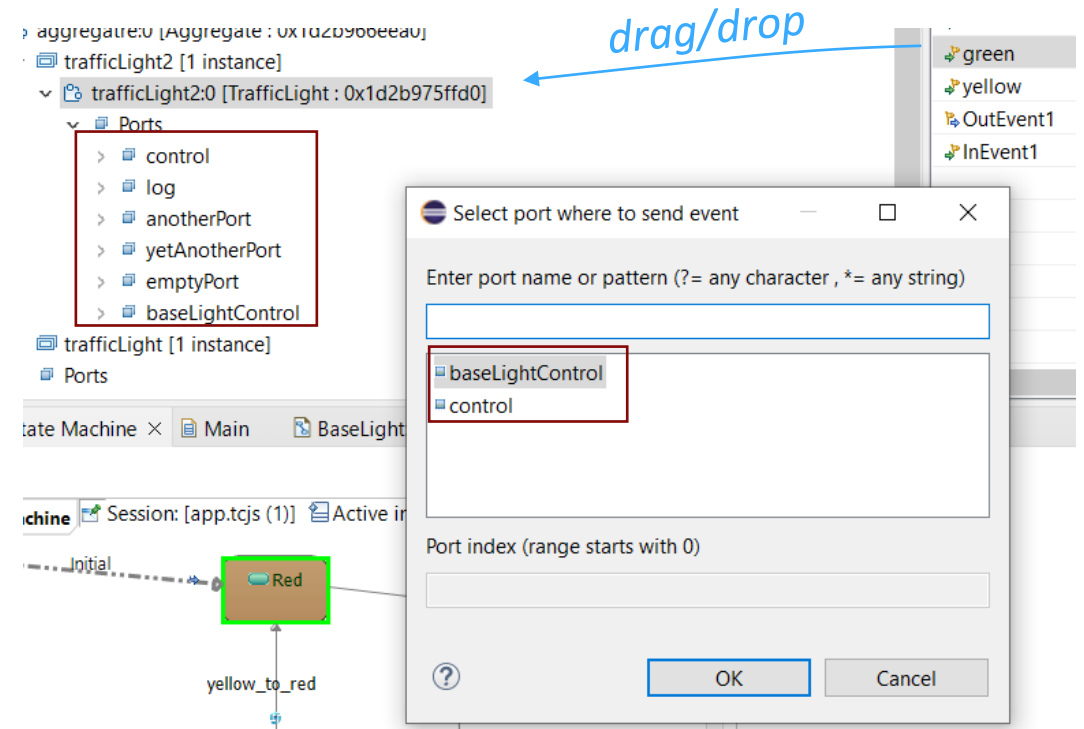
#	Trace Event	Event	Data	Action	Instance	Time
1	Event Received	red		In@control:0	trafficLight:0	621108 ms
2	Event Received	green		In@control:0	trafficLight:0	654632 ms

- ▶ The common case when tracing events sent or received on **all** ports of a capsule has been optimized
  - Accomplished by adding the capsule to the Capture tab (or selecting "Capture from all elements")
  - The trace performance is now better in case the capsule has many ports



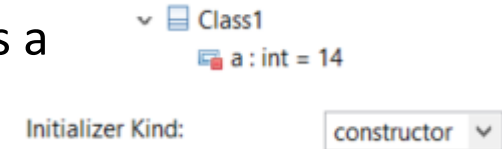
# Easier to Send Events to Ports while Debugging

- ▶ A new dialog helps to send an event to the correct port
  - Only shows ports that can receive the event to be sent
  - Appears when you drag/drop an event from the Events view onto a capsule instance in the Debug view
  - Also appears when a capsule instance is selected in the Debug view and the context menu command **Send Event** is performed in the Events view context menu.
  - Also appears when you perform the **Send Event** context menu command on a capsule instance (then only ports that can receive any event are listed and you will be prompted in a subsequent dialog for which event to send)
- ▶ If the target port is replicated (i.e. has multiplicity > 1) you can also specify the index of the port to receive the event
  - Leave the field blank to broadcast the event to all port instances



# Warning if an Attribute Default Value is Ignored During Code Generation

- ▶ A class attributes with "Initializer Kind" = "Constructor" that has a default value needs a constructor to be generated where the corresponding constructor initializer can be generated



- ▶ But it's possible (by means of other properties) to disable generation of a constructor that can have initializers. For example:

Default Constructor

**Generate**     Explicit    Inline    Default    Delete

Visibility:     public    protected    private

Default Constructor

Generate     Explicit    Inline    **Default**    Delete

Visibility:     public    protected    private

Default Constructor

Generate     Explicit    Inline    Default    **Delete**

Visibility:     public    protected    private

- In this case the attribute will obviously not be initialized as expected (unless a user-defined constructor exists)
- ▶ The model compiler now detects this inconsistency and prints a warning

**WARNING** : HelloWorld::Class1::a : This attribute has a default value and the 'Initializer Kind' property is set to 'Constructor', but no constructor will be generated where it can be initialized. The attribute default value will be ignored.

# Configuration of Model Compiler Validation Rules (1/2)

- ▶ When generating code, the model compiler checks the input model against several validation rules
- ▶ It's now possible to configure which rules to be enabled or disabled, and which severity to use in case a rule fails and a problem is reported
  - New preference **RealTime Development - Build/Transformations - C++ - Rule Configuration**
  - A similar command-line argument for the model compiler can be used for batch builds: **--ruleConfiguration**
- ▶ Each rule has a unique id (4 digits). The rule configuration is a comma-separated list of rule ids, prefixed with a letter:
  - **X**: disable the rule
  - **E**: set the rule's severity to Error
  - **W**: set the rule's severity to Warning
  - **I**: set the rule's severity to Information
- ▶ The rule id is printed right after the rule's severity. For example:
  - `WARNING[0001] : HelloWorld::Class1::a : This attribute has a default value and the 'Initializer Kind' property is set to 'Constructor', but no constructor will be generated where it can be initialized. The attribute default value will be ignored.`
  - If no id is printed it means the rule cannot (yet) be configured.

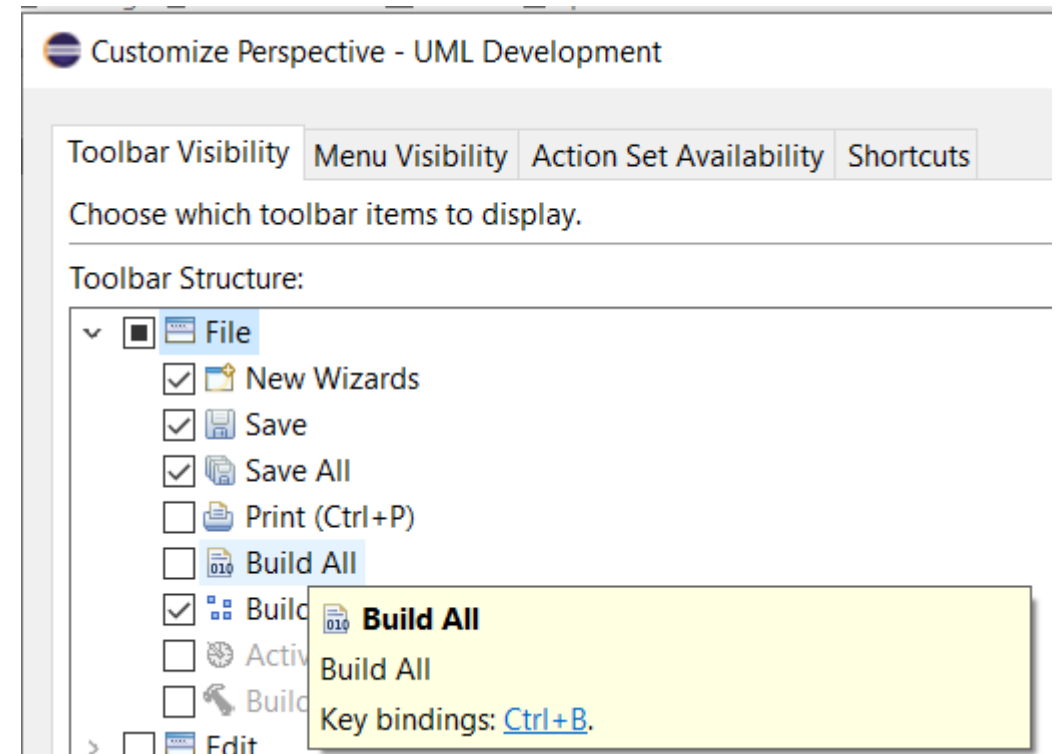
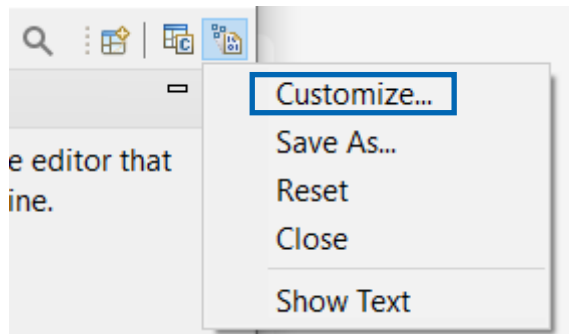
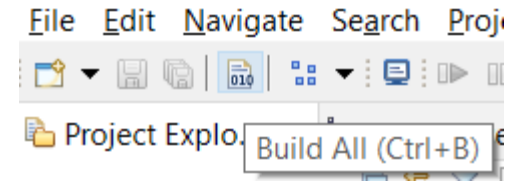
# Configuration of Model Compiler Validation Rules (2/2)

- ▶ Currently 5 validation rules can be configured:

Rule id	Rule name	Description	Default Severity
0001	AttributeInitialization	Signifies that the class does not contain a constructor where an attribute with a default value can be initialized. Hence, the attribute is ignored.	Warning
0002	TransitionUnreachable	Occurs when the model contains duplicate trigger events on more than one transition from the same origination point.	Warning
0003	EffectDuplicated	Indicates that a duplicate method has been generated as a result of the transition effect, and that the triggers are not compatible with the inherited method.	Warning
0004	GuardDuplicated	Indicates that a duplicate method has been generated for the guard of the transition as the triggers are not compatible with the inherited method.	Warning
0005	MultiplicityBadFormat	Indicates that the application is unable to parse multiplicity when an incorrect value of multiplicity is entered.	Warning

# Build All Command Removed from Toolbar

- ▶ The Build All command (contributed by CDT) has been removed from the toolbar in the UML Development perspective
  - It was too close to the "Build Active Transformation Configuration" button - easy to press the wrong button when building a TC
- ▶ With an old perspective you may need to do a Reset to get rid of the button
- ▶ If you want the button back, you can customize the UML Development perspective



# "Old-style" Code Generation for C++ 98 and Earlier

- ▶ Two new C++ code standards are now available

- **C++ 98**

- Works like "Older than C++ 11" worked before, i.e. generates code compliant with the C++ 98 standard

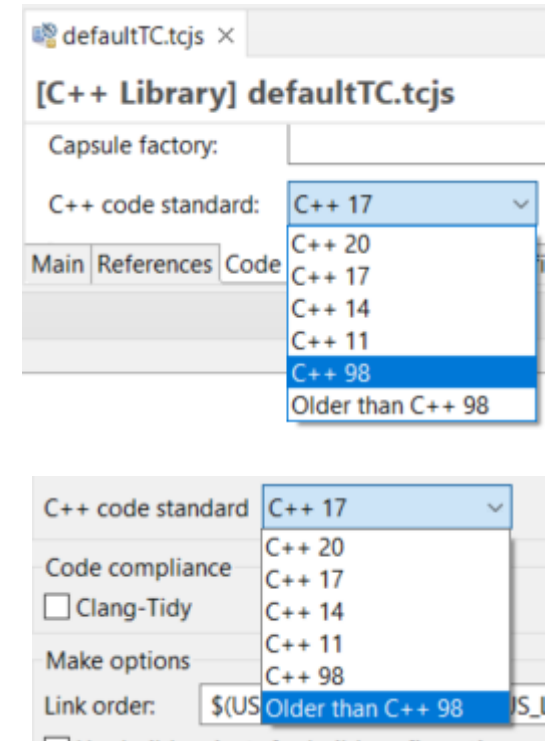
- **Older than C++ 98**

- Can be used if a very old C++ compiler is used that doesn't support all of C++ 98. For this code standard, C-style casts will be used instead of C++ casts.

- ▶ Note that for both these code standards you cannot use the latest version of the TargetRTS, since it requires C++ 11

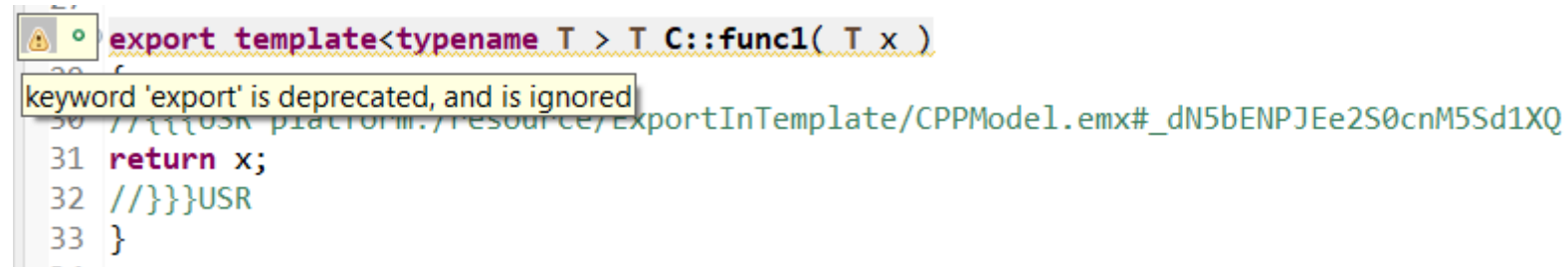
- ▶ Use these code standards only if you use an old enough TargetRTS (11.0 2020.22 or older) and need to compile generated code with an old C++ compiler

- But the benefit is that now you can use the latest version of RTist even when targeting these old environments



# Removal of "export" Keyword for Templates

- ▶ Starting from C++ 11 the `export` keyword is deprecated, and in C++ 20 it now has a different meaning
  - But even before C++ 11, very few compilers supported this language feature
  - Many modern compilers give warnings if templates use the `export` keyword.



```
export template<typename T > T C::func1( T x )  
keyword 'export' is deprecated, and is ignored  
30 //{{{OSK platform./Resource/ExportInTemplate/CPPModel.emx#_dN5bENPJEE2S0cnM5Sd1XQ  
31 return x;  
32 //}}}}USR  
33 }
```

- ▶ To avoid such warnings and be compliant with modern C++, the code generator no longer generates the `export` keyword

# Moving Event Data - Additional Constructor Generated

- ▶ The RTTypedValue struct that is generated for a user-defined type now has an additional constructor
- ▶ Allows moving (instead of copying) event data also for cases when the type descriptor needs to be explicitly provided by the user
  - For example when sending an event with data that is a subclass instance while the event parameter is typed by a superclass
- ▶ In the TargetRTS, RTTypedValue\_RTString now also has such a constructor

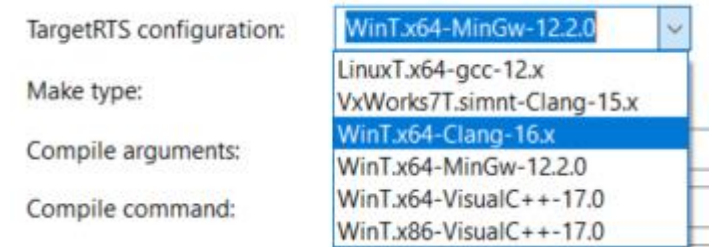
```
struct RTTypedValue_MyClass
{
    const void * data;
    const RTObject_class * type;
    const bool rValueRef = false;
    inline RTTypedValue_MyClass( const MyClass & rtg_value )
        : data( &rtg_value )
        , type( &RTType_MyClass )
    {
    }
    inline RTTypedValue_MyClass( const MyClass && rtg_value )
        : data( &rtg_value )
        , type( &RTType_MyClass )
        , rValueRef( true )
    {
    }
    inline RTTypedValue_MyClass( const MyClass & rtg_value, const RTObject_class * rtg_type )
        : data( &rtg_value )
        , type( rtg_type )
    {
    }
    inline RTTypedValue_MyClass( const MyClass && rtg_value, const RTObject_class * rtg_type )
        : data( &rtg_value )
        , type( rtg_type )
        , rValueRef( true )
    {
    }
    inline ~RTTypedValue_MyClass( void )
    {
    }
};
```

*new constructor*



# Support for the Clang Compiler

- ▶ The Clang 16.x C++ compiler for Windows (version mingw-w64-x86\_64-clang) can now be directly used with RTist
  - A target configuration for this compiler is available
  - Prebuilt libraries for this compiler are included (TargetRTS, Connexis)
  - Note that the LibTCPSTServer is not yet supported for this compiler

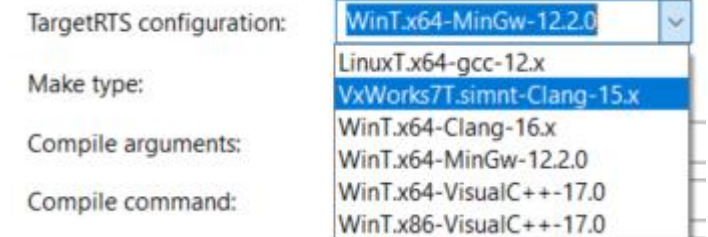


- ▶ Older versions of RTist included support for the VxWorks RTOS
  - It was temporarily removed since it didn't work with modern versions of VxWorks



- ▶ Now there is built-in support for building with the Clang 15.x compiler for VxWorks 7

- A new target configuration is available: VxWorks7T.simnt-Clang-15.x
- Prebuilt libraries for this compiler are included (TargetRTS, Connexis) and can be used for building applications to be run on the VxWorks simulator on Windows
- Note that the LibTCPserver is not yet supported for this compiler



- ▶ Learn more about this in the documentation at **RTist User's Guide - Articles - Integrations - VxWorks Integration**

**HCL**

*Relationship*<sup>TM</sup>  
BEYOND THE CONTRACT

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



WATCH THE FILM