
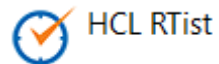


What's New in HCL RTist 11

updated for release 2020.33

Overview

- ▶ RTist 11 is based on Eclipse 2019.06 (4.12)
- ▶ HCL RTist is 100% compatible with IBM RSARTE. All features in IBM RSARTE are also present in HCL RTist. However, HCL RTist contains a few features that do not exist in IBM RSARTE.
 - Those features are marked in this presentation by 



Version: 11.0.0.v20200824_0636
Release: 2020.33

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

(c) Copyright HCL Technologies Ltd. 2016, 2020. All rights reserved.

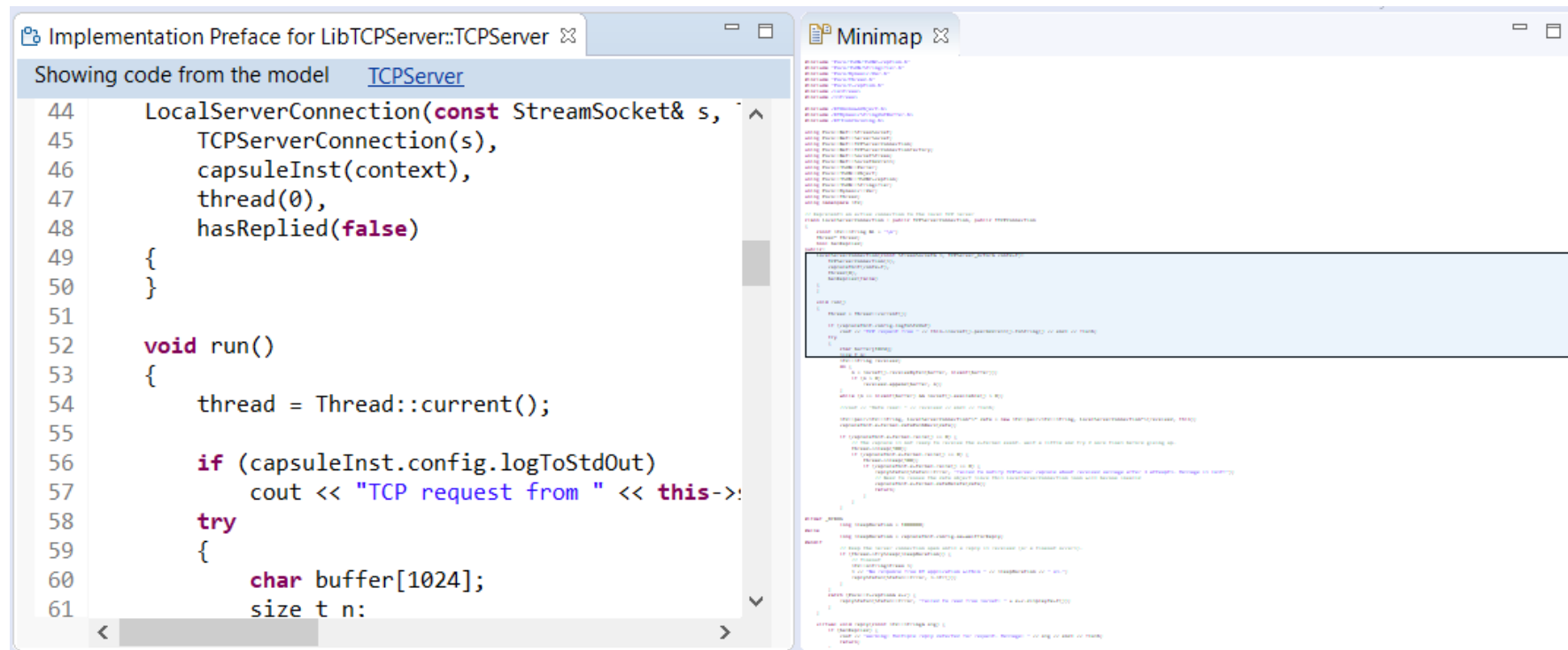
Visit <https://RTist.hcldoc.com/help/topic/com.ibm.xtools.rsarte.webdoc/users-guide/overview.html>

Eclipse 4.12 (2019.06)

- ▶ Compared to RTist 10.3, RTist 11 includes new features from 4 quarterly Eclipse releases:
 - 2018.09 (<https://www.eclipse.org/eclipse/news/4.9/platform.php>)
 - 2018.12 (<https://www.eclipse.org/eclipse/news/4.10/platform.php>)
 - 2019.03 (<https://www.eclipse.org/eclipse/news/4.11/platform.php>)
 - 2019.06 (<https://www.eclipse.org/eclipse/news/4.12/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 4.12 (2019.06)

- ▶ A new Minimap view gives a better overview of the text editor contents (useful when it's large)
 - It's easy to launch it by typing "minimap" in the QuickAccess field

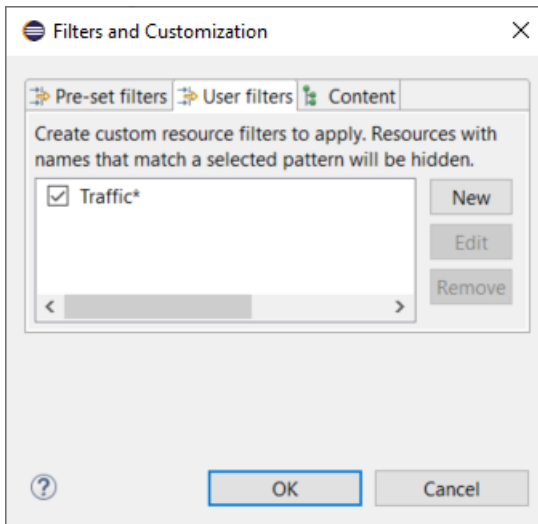
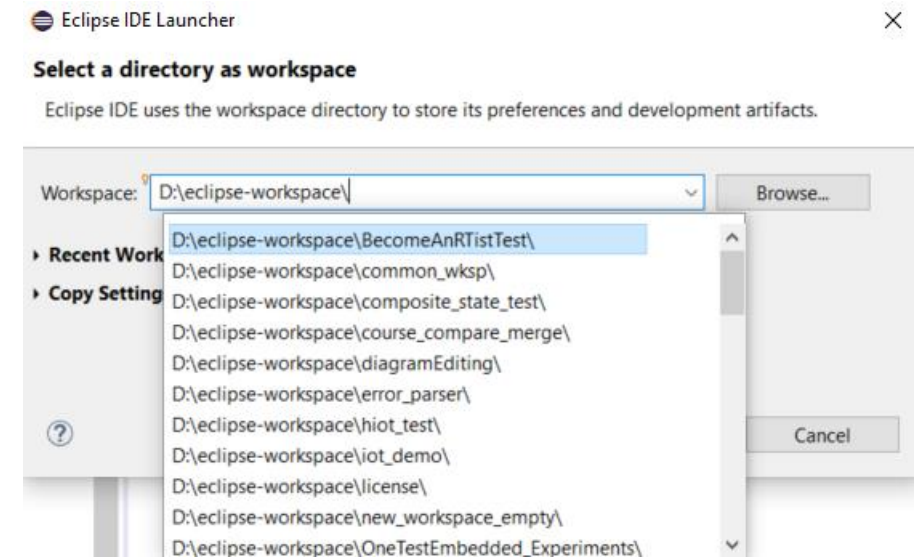


The screenshot displays the Eclipse IDE interface. On the left, the 'Implementation Preface for LibTCPServer::TCPServer' editor shows C++ code for a TCP server. The code includes a constructor and a `run()` method. On the right, the 'Minimap' view provides a high-level overview of the code, with a red box highlighting a specific section of the `run()` method.

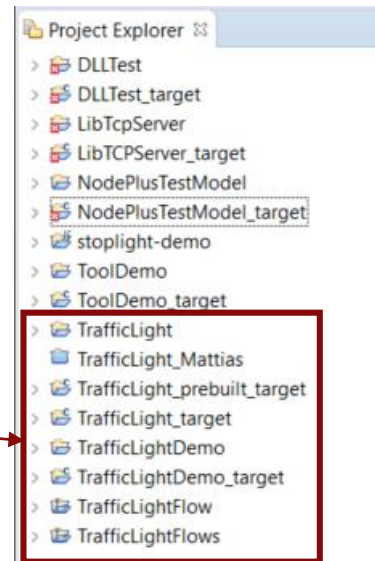
```
44 LocalServerConnection(const StreamSocket& s,  
45 TCPServerConnection(s),  
46 capsuleInst(context),  
47 thread(0),  
48 hasReplied(false)  
49 {  
50 }  
51  
52 void run()  
53 {  
54     thread = Thread::current();  
55  
56     if (capsuleInst.config.logToStdOut)  
57         cout << "TCP request from " << this->  
58     try  
59     {  
60         char buffer[1024];  
61         size_t n:
```

Eclipse 4.12 (2019.06)

- ▶ The Select Workspace dialog now supports auto-completion
 - Allows to pick a workspace more easily by only using the keyboard
- ▶ User-defined filtering of the Project Explorer
 - You can now create your own regular expressions for filtering out items from the Project Explorer
 - Can be used as an alternative to working sets



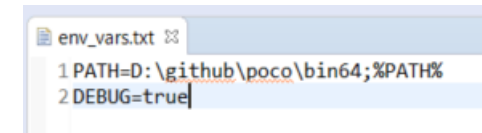
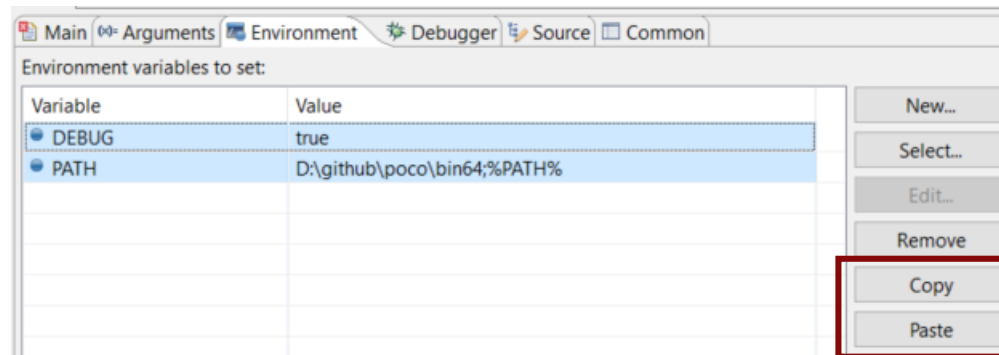
will be hidden
by the user filter



Eclipse 4.12 (2019.06)

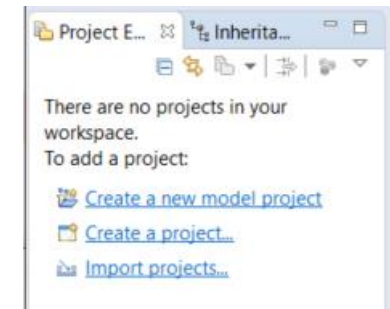
▶ Copy/Paste of environment variables

- Setting up environment variables in a launch configuration is now much easier thanks to copy/paste support
- Environment variables can be copied from one launch configuration to another, or from a text editor to a launch configuration (or vice versa)



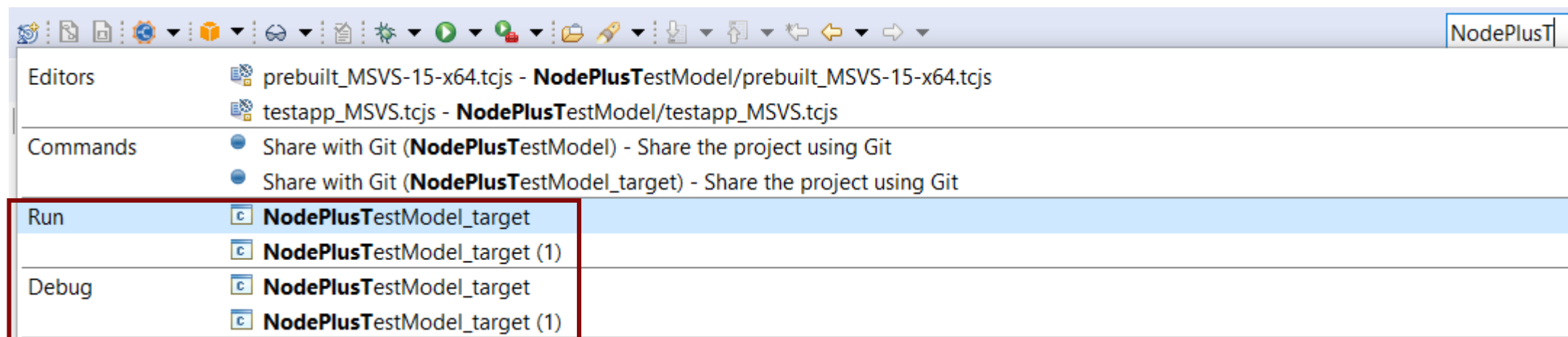
▶ Useful quick links for an empty workspace

- Quick links for creating or importing projects make it easier for new users to get started with an empty workspace
- Which quick links that are shown depend on the current perspective



Eclipse 4.12 (2019.06)

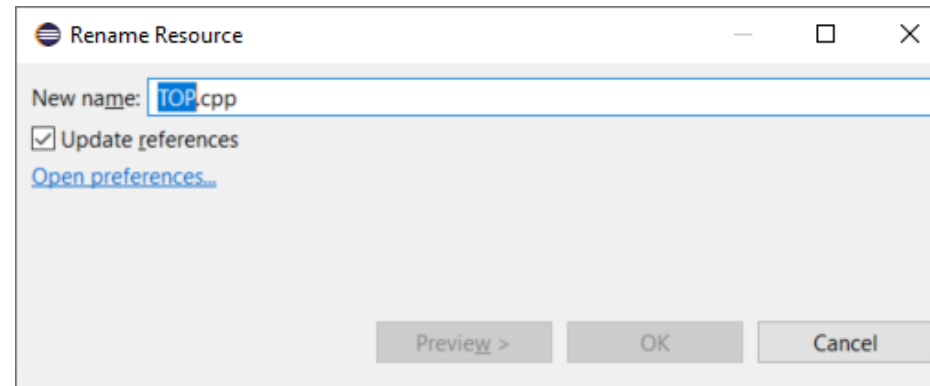
- ▶ Launch configurations are now accessible from Quick Access
 - Can start either a Run or a Debug session



CDT 9.8 (included as part of Eclipse 2019.06)

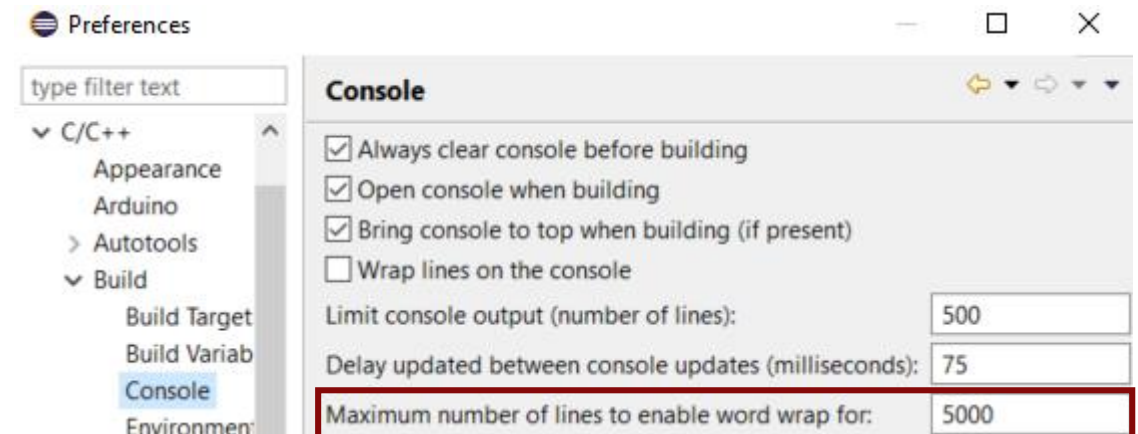
- ▶ Possible to rename a file without triggering a refactoring

- A checkbox controls if references to the renamed file should be updated



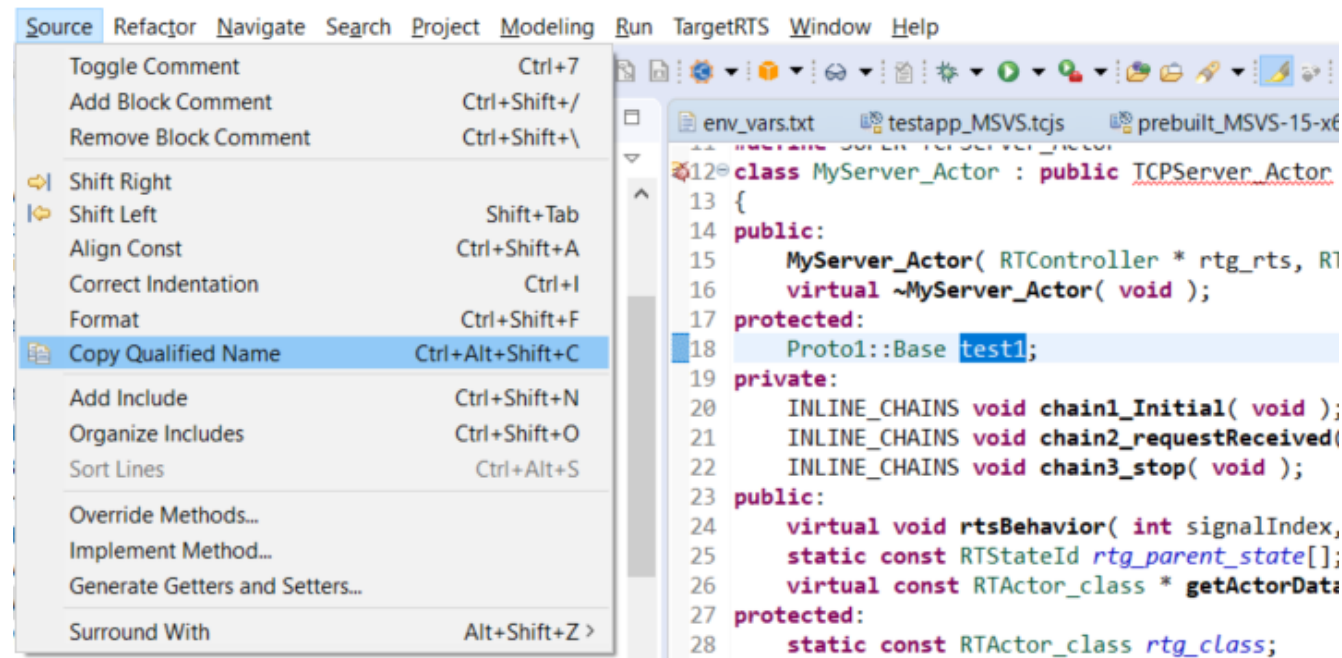
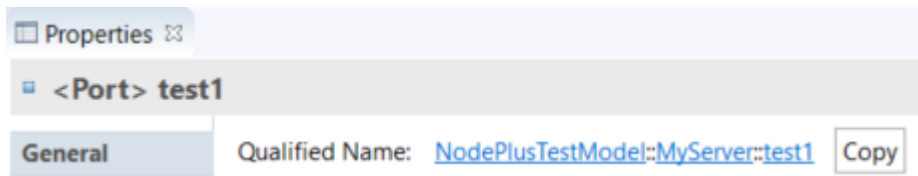
- ▶ New preference to address a performance issue with line wrapping a large number of lines in the console

- *C/C++ - Build – Console – Maximum number of lines to enable word wrap for*
- Don't set this too high if *Wrap lines on the console* is turned on



CDT 9.8 (included as part of Eclipse 2019.06)

- ▶ Improved dialog for attaching to a C++ application to debug
 - Command-line arguments for the process are now shown (allows to more easily find the process to debug)
 - The dialog now remembers a previously entered filter expression (to make it easier to attach to the same process multiple times)
- ▶ Copy the qualified name of a C++ declaration
 - A new command is available in the Source menu
 - Similar usecase as for the Copy button of the Qualified Name for a model element in the Properties view



CDT 9.8 (included as part of Eclipse 2019.06)

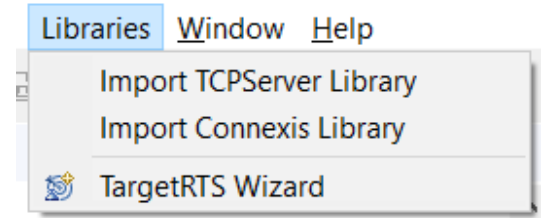
- ▶ CODAN improvements
 - Additional checks implemented, for example to detect C-style casts and goto-statements
- ▶ For more information about CDT improvements see
 - <https://wiki.eclipse.org/CDT/User/NewIn96>
 - <https://wiki.eclipse.org/CDT/User/NewIn97>
 - <https://wiki.eclipse.org/CDT/User/NewIn98>

Newer EGit Version in the EGit Integration

- ▶ The EGit integration in RTist has upgraded EGit from 5.0 to 5.4
 - This is the recommended and latest version for Eclipse 2019.06
- ▶ This upgrade provides several new features, performance improvements and bug fixes
 - For detailed information about the changes see
 - https://wiki.eclipse.org/EGit/New_and_Noteworthy/5.1
 - https://wiki.eclipse.org/EGit/New_and_Noteworthy/5.2
 - https://wiki.eclipse.org/EGit/New_and_Noteworthy/5.3
 - https://wiki.eclipse.org/EGit/New_and_Noteworthy/5.4

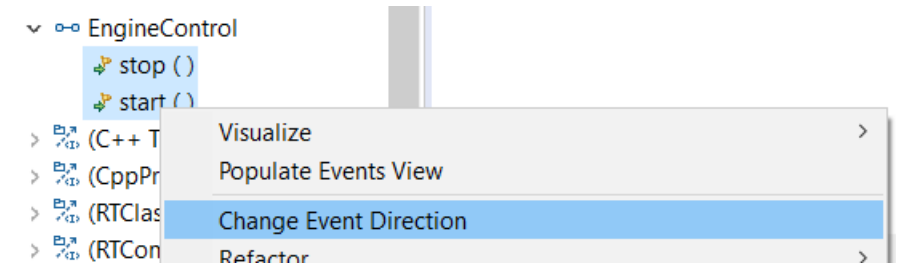
Libraries Menu

- ▶ A new menu in the menu bar with a few useful commands related to libraries
 - Contains the command for launching the TargetRTS wizard (i.e. the menu replaces the TargetRTS menu)
- ▶ Commands for importing the Connexis and TCPServer libraries
 - Available if these features have been installed
 - They import the Connexis or TCPServer library projects from the installation into the workspace



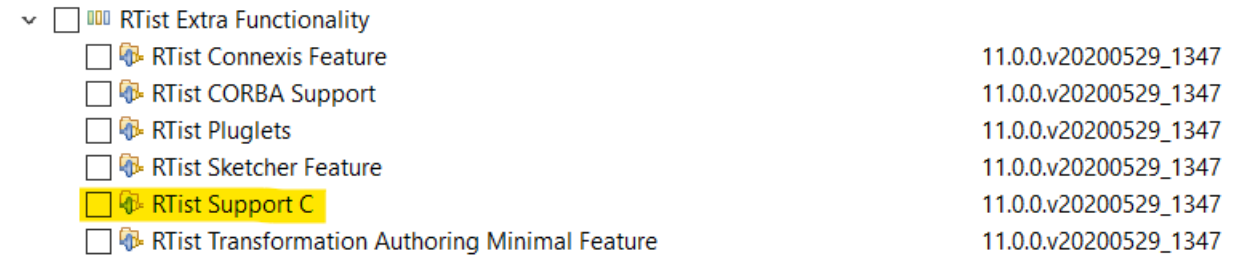
Converting Between In Events and Out Events

- ▶ It's now possible to convert an In Event to an Out Event or an Out Event to an In Event
 - This can be useful for example if you want to change the conjugation of a port, and need to swap the direction of all events in the protocol of the port
- ▶ Use the new “Change Event Direction” command in the Project Explorer context menu of a protocol event (or multiple events)



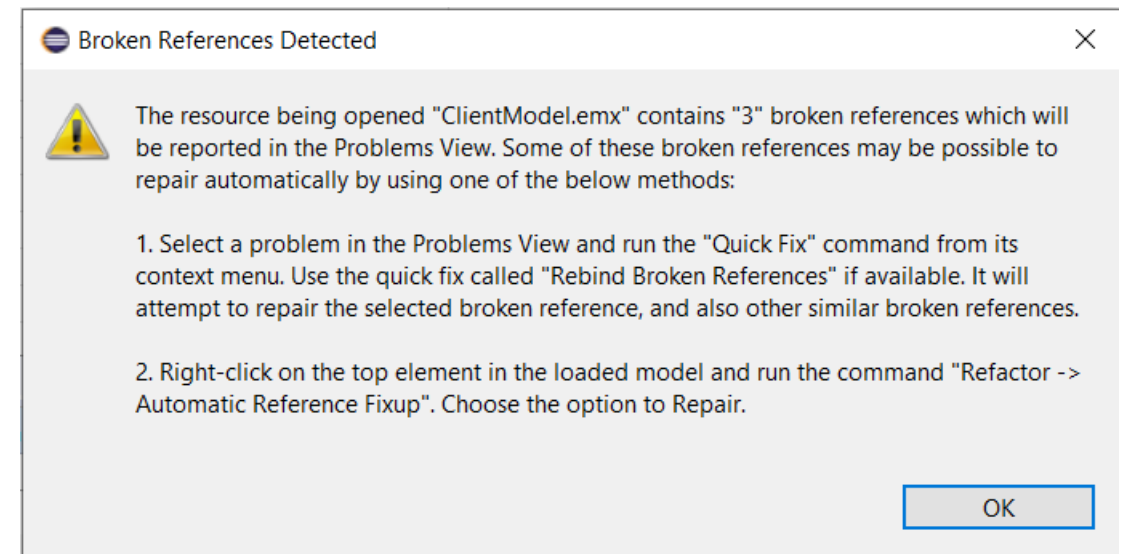
Primitive Type Improvements in Search and Type Completion

- ▶ Previously duplicates of primitive C++ types appeared when doing search and type completion
 - As a consequence it was easy to by mistake use the wrong type (and also tedious to have to choose each time)
 - The reasons for getting these duplicates have now been addressed
- ▶ The “C++ Types” package and “C++ Transformations” profile have been removed from the installation
 - They were obsolete and no longer needed
- ▶ The support for C code generation is now an optional choice when installing RTist
- ▶ New and updated model fixups are now available for updating existing models for these changes
 - Incorrect type references fixup (updated to fix references to removed C++ Types and, optionally, also C types)
 - Obsolete package import removal (new fixup)
 - Obsolete profile application removal (new fixup)
 - Add missing package imports (new fixup for adding the import of CppPrimitiveDatatypes, in case it’s missing)



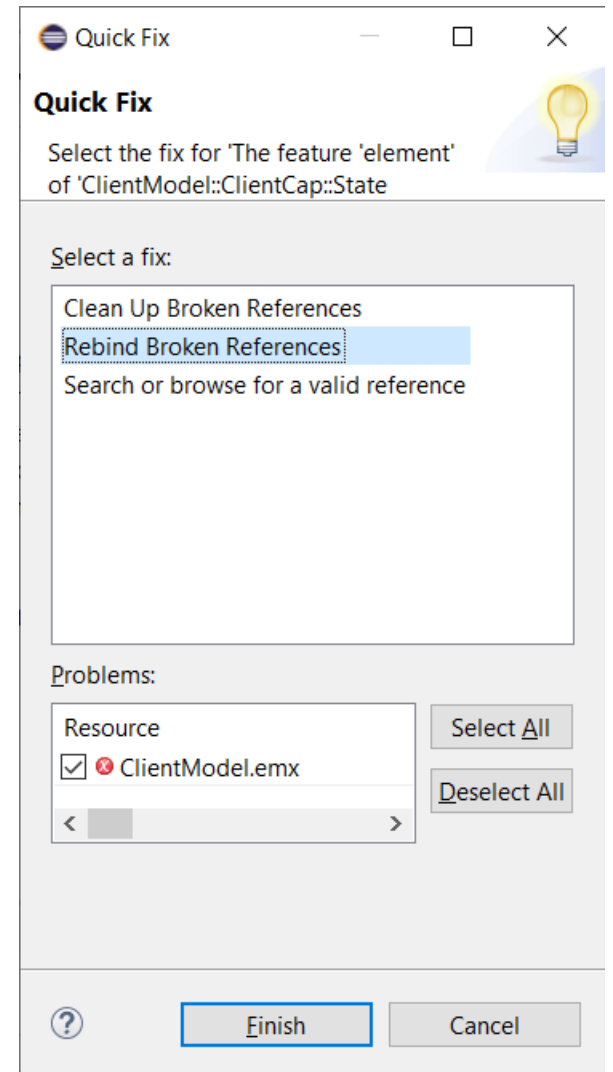
Fixing Broken References (1/2)

- ▶ The “automatic cleanup” of certain elements with broken references is now removed
 - The preference *RealTime Development - Resource Resolution - Broken references auto-cleanup* has been removed
- ▶ By default, when a model with broken references are loaded a warning dialog now appears
 - This dialog replaces the previous Repair Workspace References dialog
 - To repair the broken references, by attempting to rebind them to elements that have been moved to a different file, follow one of the methods suggested by the dialog
 - You can disable this warning dialog using the new preference *Modeling – Resource Resolution – Show warning for broken references when loading a model*
 - Note: A current limitation is that broken references caused by deleted projects will not appear automatically when loading a model. To see such broken references perform the Validate command on the loaded model.



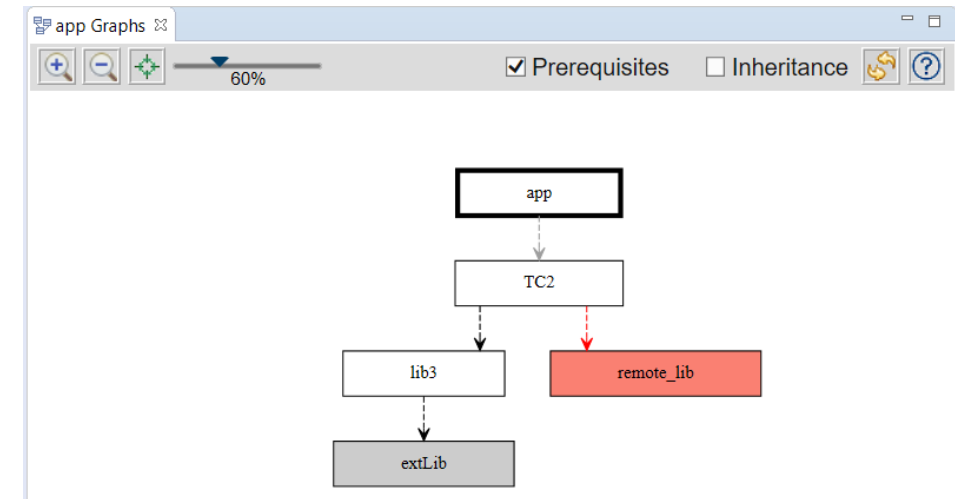
Fixing Broken References (2/2)

- ▶ Broken references are listed in the Problems view. Right-click on any of them and run a Quick Fix for fixing broken references:
 - **Rebind Broken References** can fix references to elements that have been moved. It's for example useful after a refactoring that was done without having all referencing models available in the workspace.
 - **Clean Up Broken References** can fix references to elements that have been deleted (or for some other reason do not exist in the workspace). Broken references will be fixed by resetting them (or in some cases, by deleting the element that contains the reference).
 - **Search or browse for a valid reference** can be used to rebind the broken reference to a different target element using the Select Element dialog. This Quick Fix only fixes the selected broken reference.
Note: To be able to browse to the new target element for the reference you may need to first switch to the Model viewpoint (in case the Capsule Development view point filters out the desired target element).



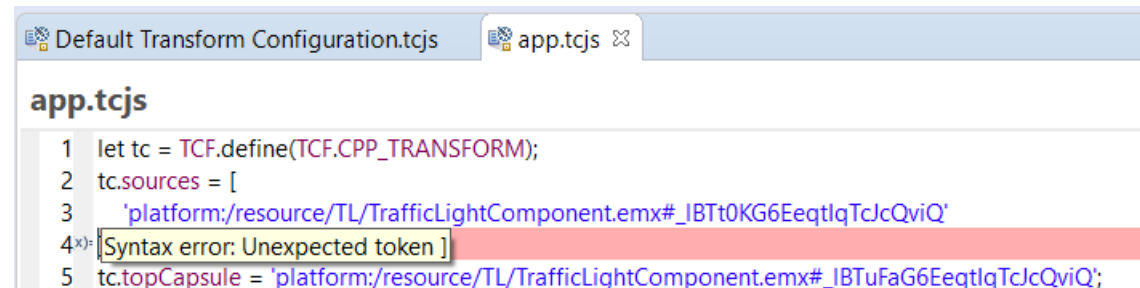
Graphic Visualization of TC Relationships

- ▶ The possibility to visualize TC relationships that was available in version 10.2 for TCs in the old file format, is now also supported for TCs in the new .tcjs file format
 - A new command **Visualize – Explore in Graphs** is available in the Project Explorer context menu of a TC
- ▶ The visualization uses Graphviz as an external tool for doing the graph layout
 - Graphviz must be separately installed. Specify the path to its “dot” utility in *Preferences – RealTime Development - Visualization*
- ▶ The graph is rendered using SVG in an embedded web browser
 - Support for panning and zooming
 - You can search for texts in the rendered diagram using Ctrl+F
- ▶ Prerequisites and/or inheritance relationships can be shown
- ▶ Double-click on TCs in the graph to open the TC editor



Transformation Configuration Editor Improvements (1/2)

- ▶ Syntax highlighting is now available in the Code tab
 - JavaScript parse errors are shown with red color and margin marker tooltip



The screenshot shows the Transformation Configuration Editor with two tabs: 'Default Transform Configuration.tcjs' and 'app.tcjs'. The 'app.tcjs' tab is active, displaying the following code:

```
1 let tc = TCF.define(TCF.CPP_TRANSFORM);
2 tc.sources = [
3   'platform:/resource/TL/TrafficLightComponent.emx#_IBTt0KG6EeqtlqTcJcQviQ'
4  ];
5 tc.topCapsule = 'platform:/resource/TL/TrafficLightComponent.emx#_IBTuFaG6EeqtlqTcJcQviQ';
```

A red horizontal bar highlights the closing bracket of the array on line 4, with a tooltip that reads "Syntax error: Unexpected token]".

- ▶ Content assist (“code completion”) is now available in the Code tab
 - Triggered automatically when typing certain “key” characters (‘.’, ‘=’ etc)
 - Can also be manually triggered using Ctrl + Space
 - Controlled by a new preference *RealTime Development – Transformation Configuration Editor – Content Assist – Enable auto activation*

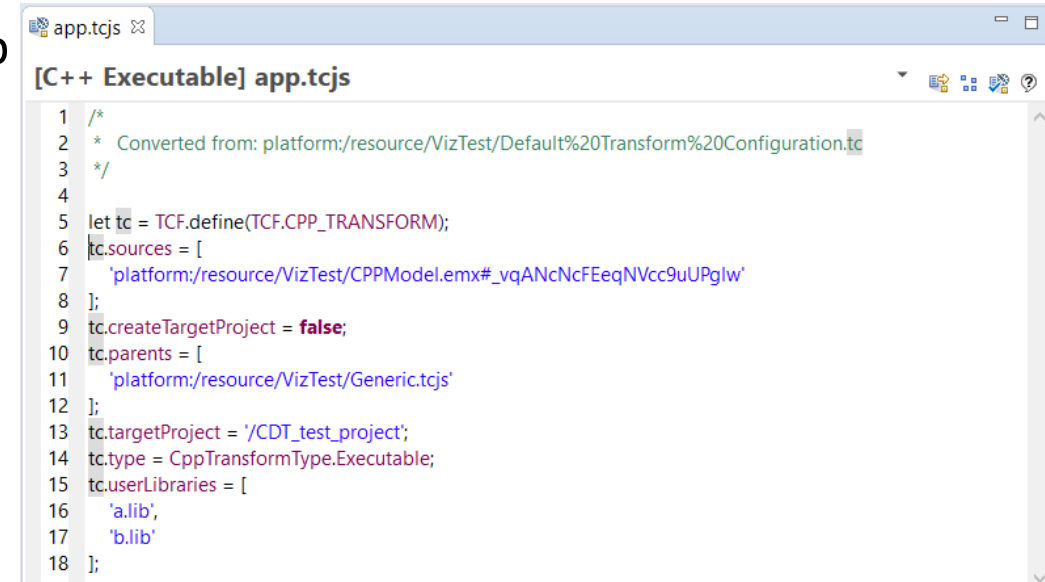


The screenshot shows the Transformation Configuration Editor with the 'app.tcjs' tab active. The code is the same as in the previous screenshot. A tooltip is visible over the 'tc.bui' property on line 5, showing the following suggestions:

- buildLibraryArguments : String
- buildLibraryCommand : String
- contextSensitiveLibraryBuild : Boolean

Transformation Configuration Editor Improvements (2/2)

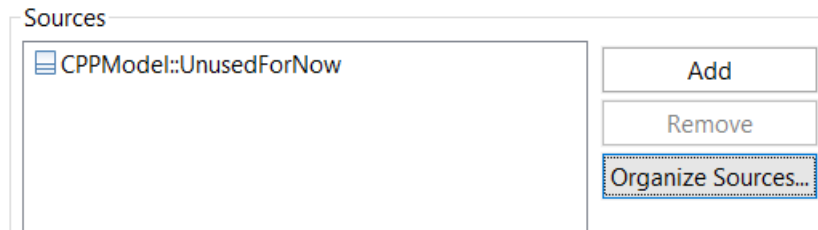
- ▶ Marking of occurrences of selected element in the Code tab
 - Controlled by a new preference *RealTime Development – Transformation Configuration Editor – Mark Occurrences – Mark occurrences of selected element*



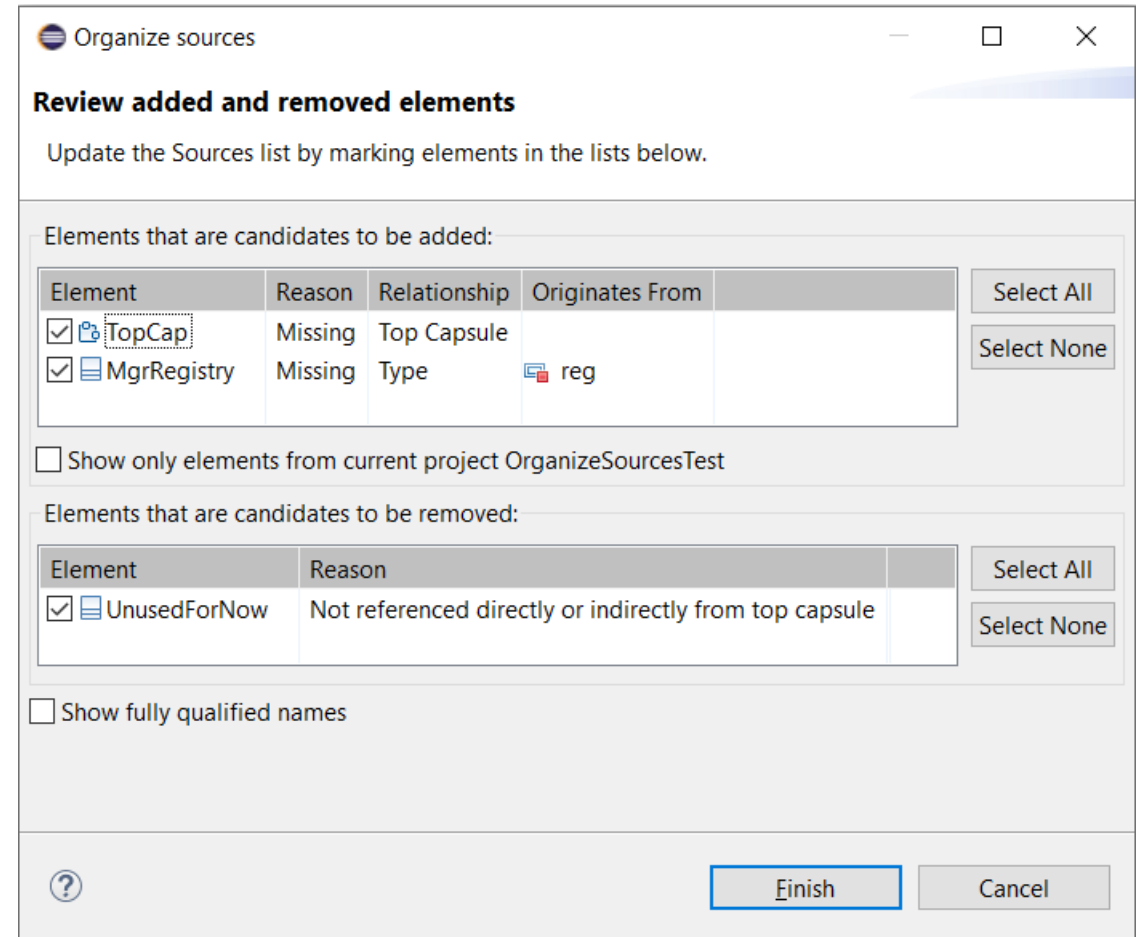
```
app.tcjs
[C++ Executable] app.tcjs
1 /*
2  * Converted from: platform:/resource/VizTest/Default%20Transform%20Configuration.tc
3  */
4
5 let tc = TCF.define(TCF.CPP_TRANSFORM);
6 tc.sources = [
7   'platform:/resource/VizTest/CppModel.emx#_vqANcNcFEeqNVcc9uUPglw'
8 ];
9 tc.createTargetProject = false;
10 tc.parents = [
11   'platform:/resource/VizTest/Generic.tcjs'
12 ];
13 tc.targetProject = '/CDT_test_project';
14 tc.type = CppTransformType.Executable;
15 tc.userLibraries = [
16   'a.lib',
17   'b.lib'
18 ];
```

Organize Sources

- ▶ A new Organize Sources dialog is provided to assist in setting up the Sources of a TC optimally



- This is an improved and simplified implementation compared to the similar feature in 10.2
- Elements are proposed to be added to or removed from the Sources list to ensure that no unnecessary elements will be built by the TC. The suggestions are made by analyzing references in the model.
- This feature is also available from the command-line by means of a new model compiler command-line argument `--organizeSources`
- It can also be launched from the TC Editor toolbar menu (useful for TCs edited only textually in the Code tab)



Cleaning TCs

- ▶ External Library TCs have a new "Clean Command" property
 - Use for example "make clean", invoke a "clean script" or something else
- ▶ When it has been specified it's possible to use the Clean command also on external library TCs
 - Currently this only works when cleaning from the command-line (using the makefile generated by the model compiler)
- ▶ When build variants are used, the Clean dialog now shows the Build Variants user interface
 - Allows setting the build variant settings which may influence on the behavior of clean, and clean build

Generate make file for external CDT project

Build command:

Build folder:

Clean command:

Clean selected transformation configurations

build.tcjs

Build Variants

Configuration:

Target platform Print TCJS name

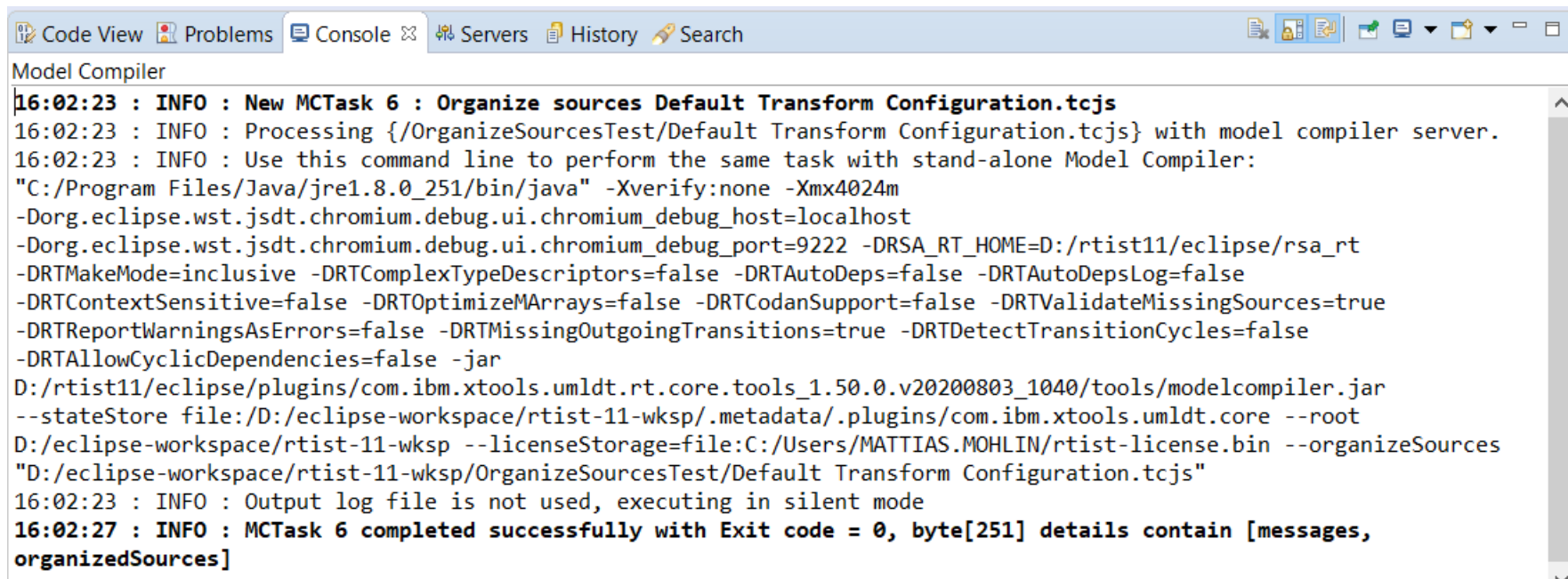
Clean binaries only

Build selected transformation configuration after clean

Set as Active Build Variants Configuration

Model Compiler Console

- ▶ A new "Model Compiler" console is now available
 - Helps troubleshooting commands that involves calls to the model compiler
 - Makes it easier to know how a model-compiler command should be invoked from the command-line

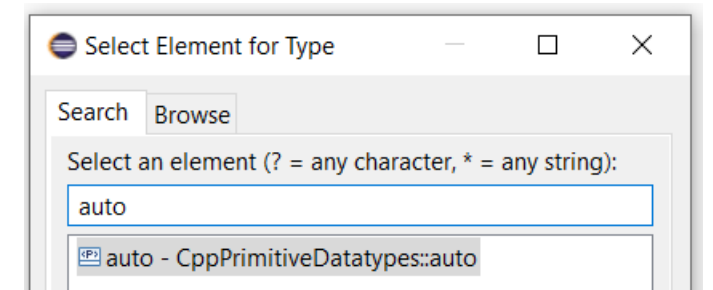
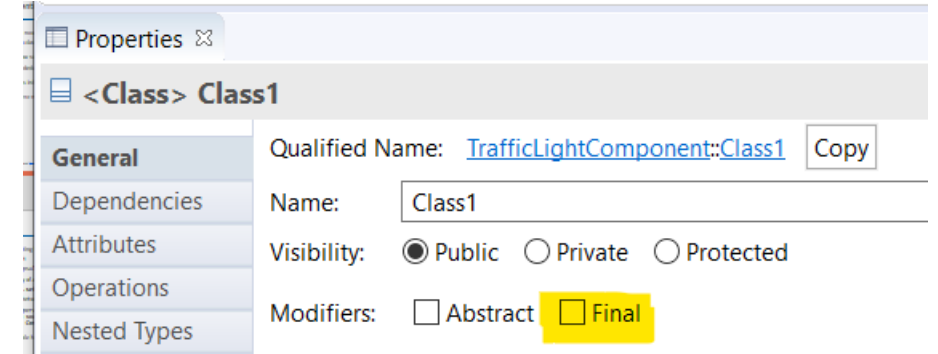


The screenshot shows the Eclipse IDE's Console window. The title bar includes tabs for Code View, Problems, Console, Servers, History, and Search. The console output is as follows:

```
Model Compiler
16:02:23 : INFO : New MCTask 6 : Organize sources Default Transform Configuration.tcjs
16:02:23 : INFO : Processing {/OrganizeSourcesTest/Default Transform Configuration.tcjs} with model compiler server.
16:02:23 : INFO : Use this command line to perform the same task with stand-alone Model Compiler:
"C:/Program Files/Java/jre1.8.0_251/bin/java" -Xverify:none -Xmx4024m
-Dorg.eclipse.wst.jsdt.chromium.debug.ui.chromium_debug_host=localhost
-Dorg.eclipse.wst.jsdt.chromium.debug.ui.chromium_debug_port=9222 -DRSA_RT_HOME=D:/rtist11/eclipse/rsa_rt
-DRTMakeMode=inclusive -DRTComplexTypeDescriptors=false -DRTAutoDeps=false -DRTAutoDepsLog=false
-DRTContextSensitive=false -DROptimizeMArrays=false -DRTCodanSupport=false -DRTValidateMissingSources=true
-DRTReportWarningsAsErrors=false -DRTMissingOutgoingTransitions=true -DRTDetectTransitionCycles=false
-DRTAllowCyclicDependencies=false -jar
D:/rtist11/eclipse/plugins/com.ibm.xtools.umldt.rt.core.tools_1.50.0.v20200803_1040/tools/modelcompiler.jar
--stateStore file:/D:/eclipse-workspace/rtist-11-wksp/.metadata/.plugins/com.ibm.xtools.umldt.core --root
D:/eclipse-workspace/rtist-11-wksp --licenseStorage=file:C:/Users/MATTIAS.MOHLIN/rtist-license.bin --organizeSources
"D:/eclipse-workspace/rtist-11-wksp/OrganizeSourcesTest/Default Transform Configuration.tcjs"
16:02:23 : INFO : Output log file is not used, executing in silent mode
16:02:27 : INFO : MCTask 6 completed successfully with Exit code = 0, byte[251] details contain [messages,
organizedSources]
```

C++ 11 Improvements (1/2)

- ▶ The **final** modifier can now be used on classes, capsules and operations
 - Use it to specify that a class/capsule cannot be inherited from, or an operation cannot be redefined
- ▶ **auto** attributes are now supported by the model compiler
 - The C++ compiler will deduce the type of the corresponding member variable from its initializer value
 - Type completion now works for auto, and it can be selected as any other type
- ▶ Attributes with in-class initialization (brace or equal) are no longer also initialized in the constructor
- ▶ Some C++ 11 types were missing in the CppPrimitiveDatatypes package and have now been added
 - long long
 - unsigned long long

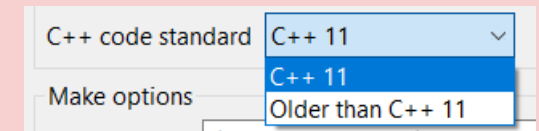


C++ 11 Improvements (2/2)

- ▶ The **override** keyword is now generated for functions that redefine virtual functions from the TargetRTS

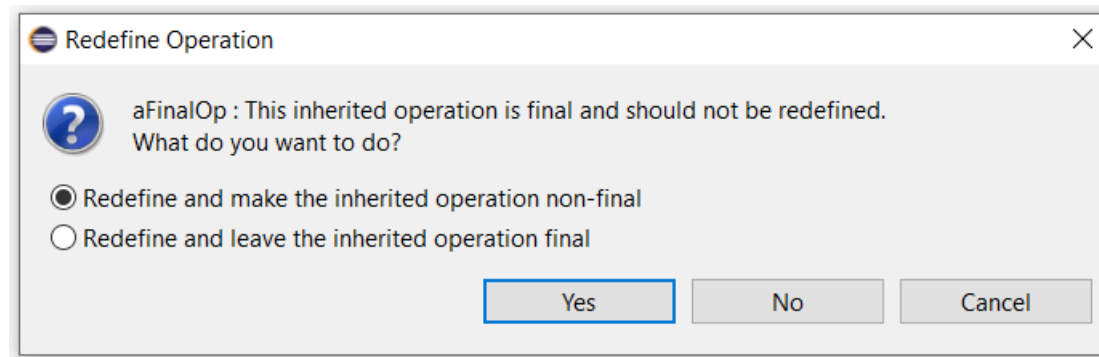
- Avoids warnings when compiling generated code

- **Note:** Starting with this change it's now required to use a C++ 11 compiler for compiling generated code. You can use the new preference *RealTime Development – Build/Transformation – C++ code standard* if you need to compile generated code with a compiler that doesn't support C++ 11.



- ▶ When overriding an operation in the UI, its 'override' property is automatically set

- If the overridden operation is final, a new dialog appears and suggests to make it non-final



Properties View Usability Improvements

- ▶ Several improvements have improved the usability of the Properties view. Some examples:
 - Properties not applicable in certain contexts are now disabled
 - Properties not applicable for RT modeling are now hidden when using the Capsule Development viewpoint
 - The “Open Type” button is now available for operation return types

Return Type:

- The terminology of some properties was improved to be aligned with C++ terminology
- Some properties were present on multiple property pages. Now those duplicates have been removed to save space.

Help for the Properties View

- ▶ The Properties View now provides context sensitive help
 - Use the new toolbar button for showing help for the properties currently visible in the view (or press F1)
 - The new property documentation helps understand what properties mean, and how they affect the generated code

The screenshot displays the Properties View for a model element named 'TopCap' under the 'Capsule' package. The 'C++ General' property is selected, and its documentation is shown in a help window. The documentation includes sections for 'Generate File', 'Header Preface', 'Header Ending', 'Implementation Preface', and 'Implementation Ending'. The 'Generate File' section explains that by default, both an implementation and header file are generated, and checkboxes are provided to suppress one or both. The other sections describe the placement of code snippets in the generated files.

The Properties View shows the following settings for the 'C++ General' property:

- Generate File: Implementation Header
- Header Preface: `1 // Some necessary includes`
`2 #include <iostream>`
`3 #include <string>`
- Header Ending: `1`
- Implementation Preface: `1`

The help window shows the following documentation for the 'C++ General' property:

- Generate File**
By default both an implementation and header file is generated for the model element. You can unmark these checkboxes to suppress generation of one or both of these files.
- Header Preface**
This code snippet will be inserted at the beginning of the C++ header file that is generated for the selected element.
- Header Ending**
This code snippet will be inserted at the end of the C++ header file that is generated for the selected element.
- Implementation Preface**
This code snippet will be inserted at the beginning of the C++ implementation file that is generated for the selected element.
- Implementation Ending**

UML-RT Java APIs

- ▶ The Java APIs for creating UML-RT models programmatically have been extended

- Creating a capsule part

```
RTCapsulePart RTCapsule.createCapsulePart(String name, RTCapsule type);
```

- Setting the multiplicity of a capsule part, and its kind (fixed, optional or plugin)

```
void RTCapsulePart.setMultiplicityAndKind(int lower, int upper, AggregationKind kind);
```

- Creating a connector between two ports on capsule parts

```
Connector RTModelOperations.createConnector(RTCapsulePart sourcePart,  
RTPortRedefinition sourcePort, RTCapsulePart targetPart, RTPortRedefinition  
targetPort) throws ConnectionError;
```

- ▶ The UML RT SDK sample has been updated to use these new APIs

Usage of CDATA when Storing Models

- ▶ A new preference can be set to store code snippets and documentation texts using CDATA in the model XML files
- ▶ This avoids use of XML escape characters which makes such texts more readable when viewed in a text editor

■ Example (without CDATA)

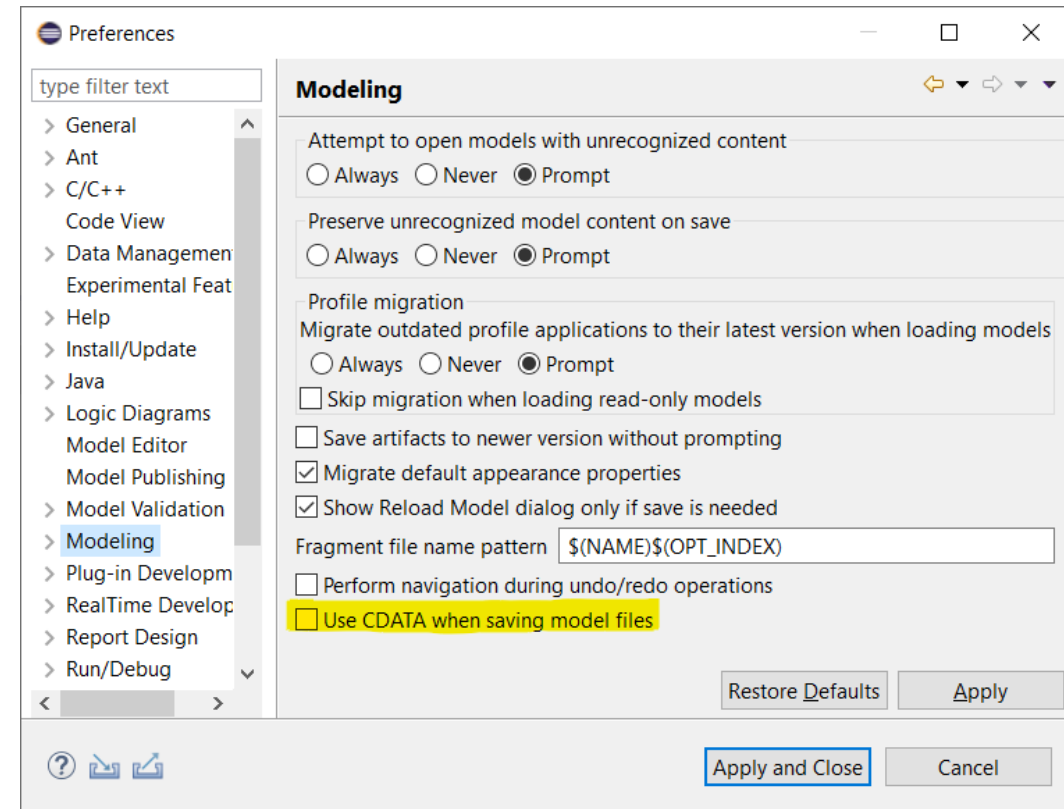
```
<headerPreface xmi:id="_irLY4NitEeqes4xYsqINtA"  
body="// Some necessary  
includes&#xD;&#xA;#include  
&lt;iostream>&#xD;&#xA;#include  
&lt;string>&#xD;&#xA;&#xD;&#xA;#define OPEN  
1&#xD;&#xA;#define CLOSED 0"/>
```

■ Example (with CDATA)

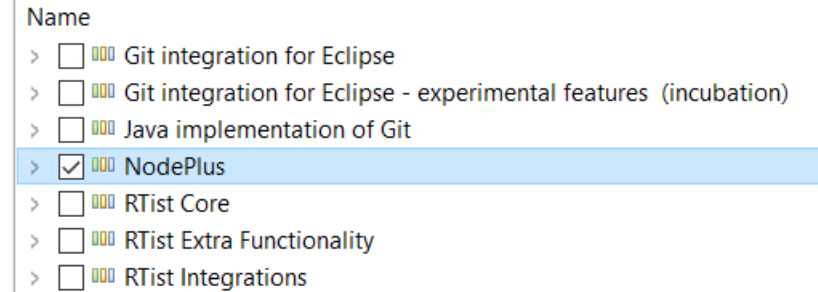
```
<headerPreface xmi:id="_irLY4NitEeqes4xYsqINtA">  
  <body><![CDATA[// Some necessary includes  
#include <iostream>  
#include <string>  
  
#define OPEN 1  
#define CLOSED 0]]></body>  
</headerPreface>
```

Note: Use of CDATA does not affect backwards compatibility.

Old versions of RTist can still load model files that use CDATA. But if saved in old tool versions, CDATA will of course not be used. A new model fixup is available for converting all model files to use the new file format with CDATA.

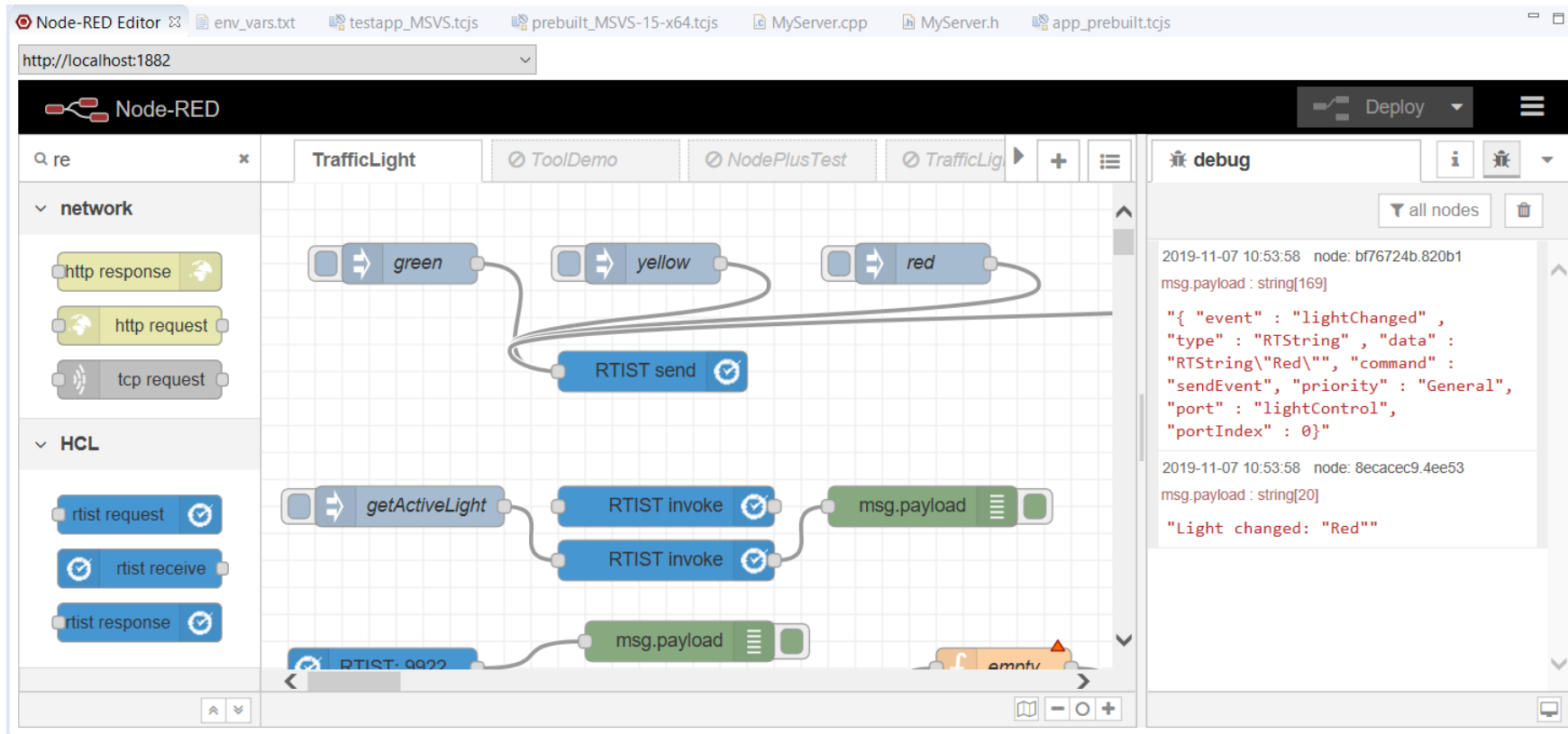


- ▶ HCL NodePlus is a new install component of HCL RTist
 - Note: Currently NodePlus is not available in IBM RSARTE
- ▶ NodePlus provides all necessary tooling for developing IoT applications using Node-RED (and related technologies)
 - Node-RED projects - both for creating Node-RED nodes and flows of interconnected nodes
 - Node-RED server - for convenient deployment of Node-RED flows within the Eclipse workbench
 - Node.js projects - for creating Node.js applications
 - Node.js server - for convenient deployment of Node.js applications within the Eclipse workbench
 - Swagger - for specifying APIs using Swagger documents
 - Node.js Debugger - for debugging Node.js applications
 - Pug (formerly known as Jade) – for template-based rendering of web pages
 - Unit test
 - Code coverage



▶ Node-RED editor

- Configure nodes and wire them together into flows
- View debug output and execution within the flow
- Deploy changes to the Node-RED server
- Install new nodes to the palette (e.g. from the public library)

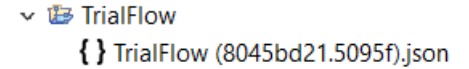


The screenshot displays the Node-RED editor interface. The main workspace shows a flow for a traffic light control system. The flow starts with three nodes labeled 'green', 'yellow', and 'red' in the top row. These nodes are connected to an 'RTIST send' node. Below this, there is a 'getActiveLight' node connected to two 'RTIST invoke' nodes, which are then connected to a 'msg.payload' node. The 'debug' console on the right shows two messages:

```
2019-11-07 10:53:58 node: bf76724b.820b1  
msg.payload : string[169]  
  
{"event": "lightChanged",  
"type": "RTString", "data":  
"RTString\\Red\\", "command":  
"sendEvent", "priority": "General",  
"port": "lightControl",  
"portIndex": 0}  
  
2019-11-07 10:53:58 node: 8ecacec9.4ee53  
msg.payload : string[20]  
"Light changed: \"Red\""
```

▶ Node-RED Flow Projects

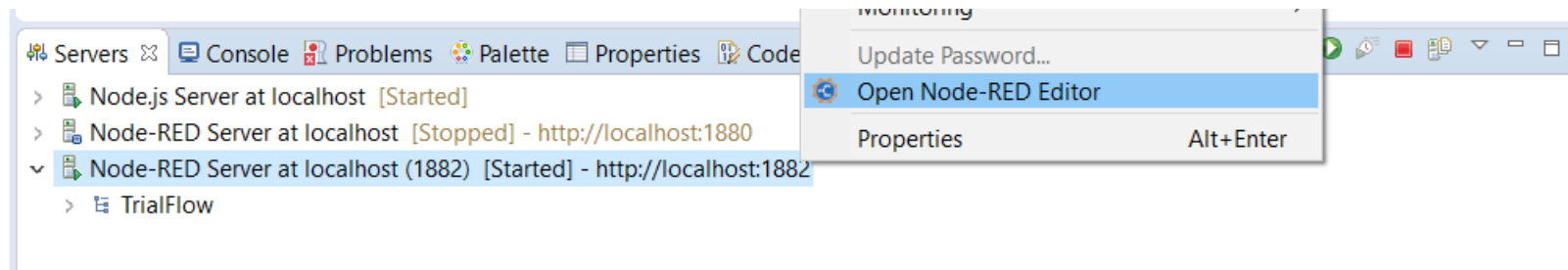
- Flows can be created directly in the Node-RED editor, but creating them as Eclipse projects has many benefits (e.g. team development using SCM integration)
- An intuitive wizard for creating Node-RED Flow projects
- Either create a flow from scratch or start from one of the existing flows in the public library



▼ TrialFlow
 { } TrialFlow (8045bd21.5095f).json

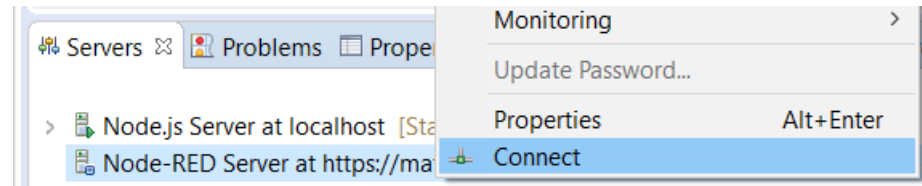
▶ Node-RED Server

- A dedicated Servers view lets you create, start and stop Node-RED servers within the Eclipse workbench
- Deploy a Node-RED flow onto a selected server and open its Node-RED editor



▶ Remote Node-RED Server

- You don't need to run the Node-RED server locally. You can instead connect to a server that runs remotely (for example on a server or on a device)
- Create a new server in the Servers view and enter the URL to the remote server
- You can then connect to the server in order to open the Node-RED editor for it (other operations on a remote server are currently not supported).

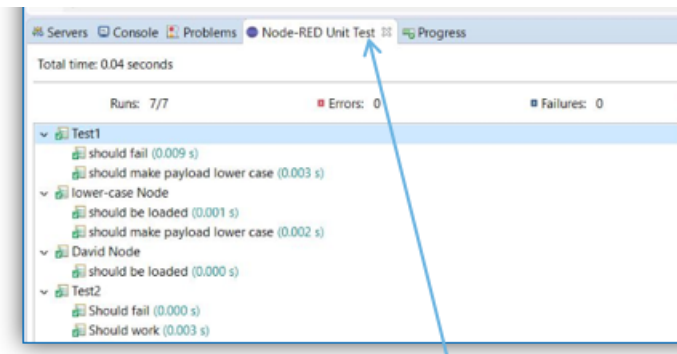
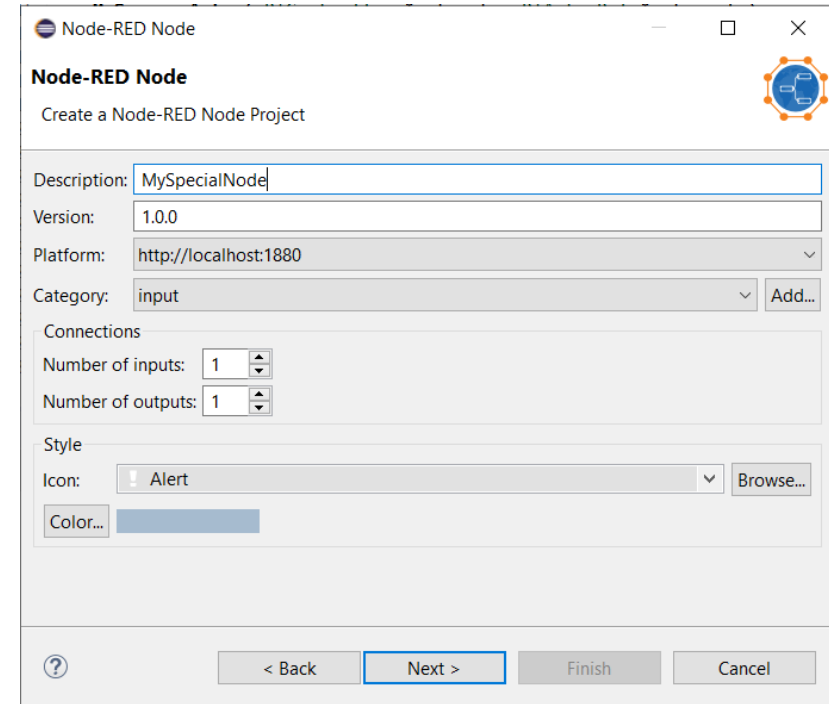


▶ Node-RED Node Projects

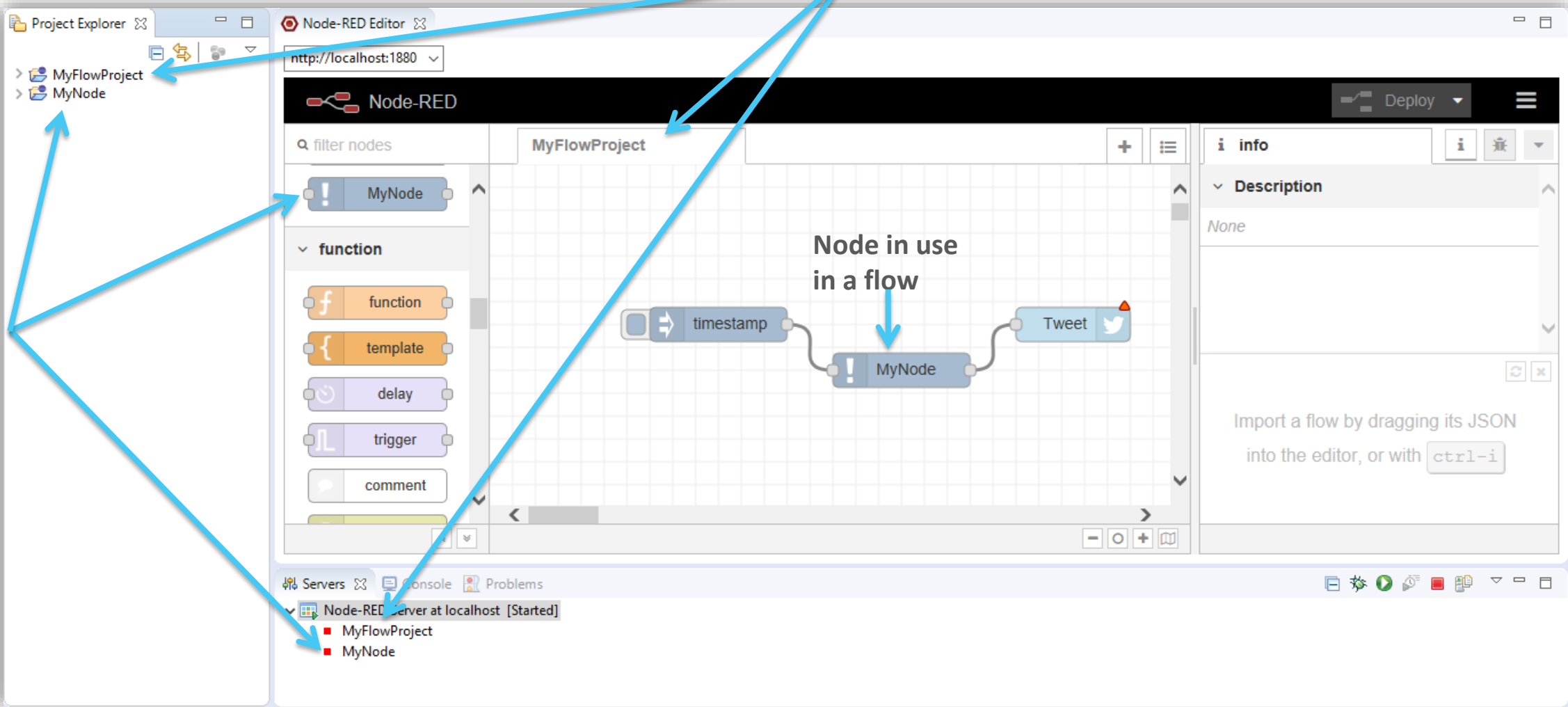
- More than 2500 nodes exist in the public library, but sometimes you may need to create your own specific node
- An intuitive wizard for creating Node-RED Node projects
- Benefit from all the usual Eclipse features when developing your node (e.g. SCM integration)
- Either create a node from scratch or start from one of the existing nodes in the public library

▶ Unit test of Node-RED nodes

- Unit tests are created using the Mocha framework and results are shown in a special Node-RED Unit Test view



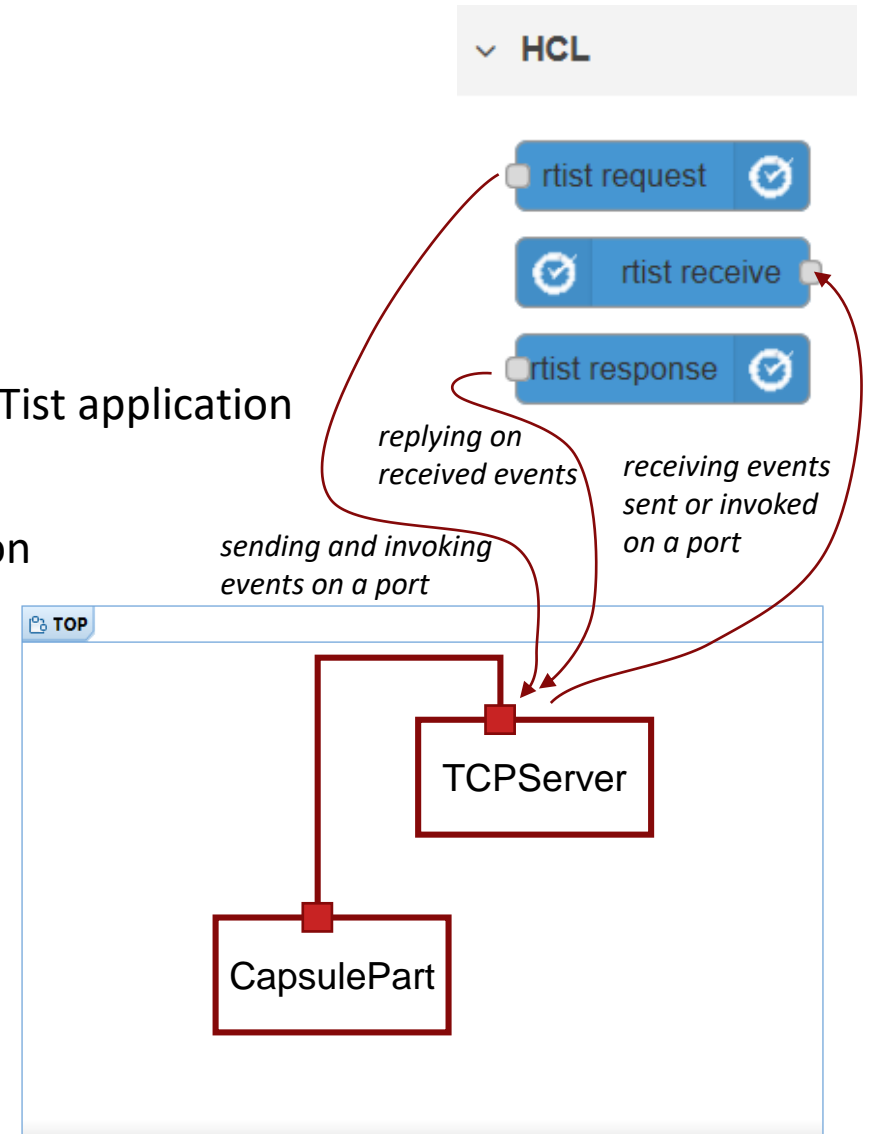
Custom "Unit Test" view



The screenshot shows the Node-RED Editor interface within the NodePlus IDE. The interface is divided into several panels:

- Project Explorer (Left):** Shows a tree view with folders for `MyFlowProject` and `MyNode`. A blue arrow labeled "Node project" points to this area.
- Node-RED Editor (Center):** The main workspace showing a flow project named `MyFlowProject`. The flow contains three nodes: `timestamp`, `MyNode`, and `Tweet`. A blue arrow labeled "Flow project" points to the top of the editor. Another blue arrow labeled "Node in use in a flow" points to the `MyNode` node in the flow.
- Node Palette (Left):** A list of available nodes, including `MyNode` (with a warning icon), `function`, `template`, `delay`, `trigger`, and `comment`. A blue arrow labeled "Node project" points to this palette.
- Info Panel (Right):** Displays information for the selected `MyNode`, including a description field (currently empty) and instructions on how to import a flow by dragging its JSON or using `ctrl-i`.
- Servers Panel (Bottom):** Shows a `Node-RED Server at localhost [Started]` with sub-items for `MyFlowProject` and `MyNode`. A blue arrow labeled "Node project" points to this panel.

- ▶ Nodes for communicating with an RTist application
 - **rtist request**
Let a flow send or invoke an event into an RTist application
 - **rtist receive**
Trigger a flow when receiving an event that is sent or invoked from an RTist application
 - **rtist response**
Reply to a received event that is sent or invoked from an RTist application
- ▶ These nodes make use of the JSON API provided by [lib-tcp-server](#) (i.e. the RTist application must use this library)
 - The nodes can be statically configured in the Node-RED editor
 - It's also possible to provide many of the node properties in the message payload for a more dynamic behavior



Swagger Support in NodePlus



► Document APIs using Swagger in a new Swagger editor

Link source and design panes

Support for JSON and YAML syntax

The screenshot shows the Swagger editor interface. On the left, a code editor displays a JSON file named 'swagger.json' with syntax highlighting and content assist. The JSON content is as follows:

```
1 "license": {
2   "name": "Apache 2.0",
3   "url": "http://www.apache.org/licenses/LICENSE-2.0.html"
4 },
5 "termsOfService": "http://swagger.io/terms/",
6 "title": "Swagger Petstore",
7 "version": "1.0.0"
8 },
9 "host": "petstore.swagger.io",
10 "basePath": "/v2",
11 "schemes": [
12   "https"
13 ],
14 "paths": {
15   "/pet": {
16     "post": {
17       "consumes": [
18         "application/json",
19         "application/xml"
20       ],
21       "parameters": [
22         {
23           "description": "Pet object that needs to be added to the store",
24           "in": "body",
25           "name": "body",
26           "required": true,
27           "schema": {
28             "$ref": "#/definitions/Pet"
29           }
30         }
31       ],
32       "responses": {
33         "405": {
34           "description": "Invalid input"
35         }
36       },
37       "security": [
38         {
39           "petstore_auth": [
40             "write:pets",
41             "read:pets"
42           ]
43         }
44       ]
45     }
46   }
47 },
48 "tags": [
```

On the right, the visual API representation for the 'Swagger Petstore' is shown. It includes fields for Version (1.0.0), Description, Terms of service, License, and Documentation. Below these are fields for Base URL (petstore.swagger.io/v2) and Schemes (https). The 'Methods' section is expanded to show the '[post] /pet' operation, which is described as 'Add a new pet to the store'. The API operation details include Name (body), Description (Pet object that needs to be added to the store), Location (body), and Required (checked). An example JSON response is shown: { photoUrls: ["string"], name: "doggie", id: "0", category: { name: "string", id: "0" } }. The 'Request type' is set to 'application/json' and the 'Response type' is empty. A table at the bottom shows a 405 status code with the description 'Invalid input'.

Visual API representation

Support for executing / testing API

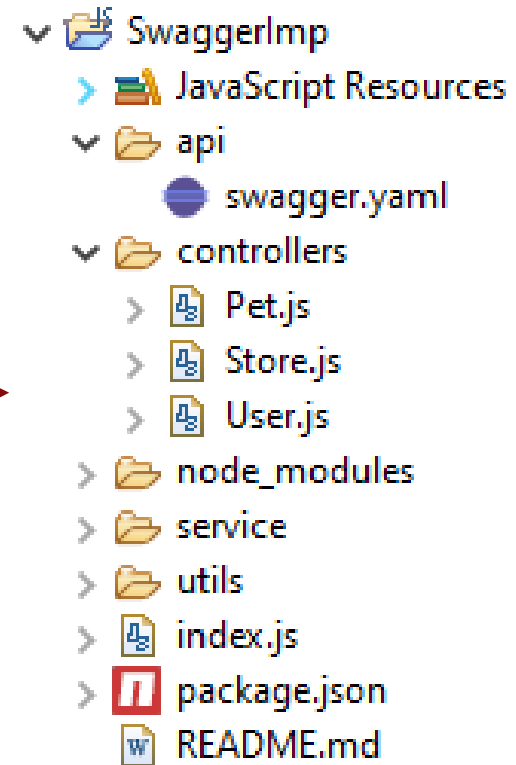
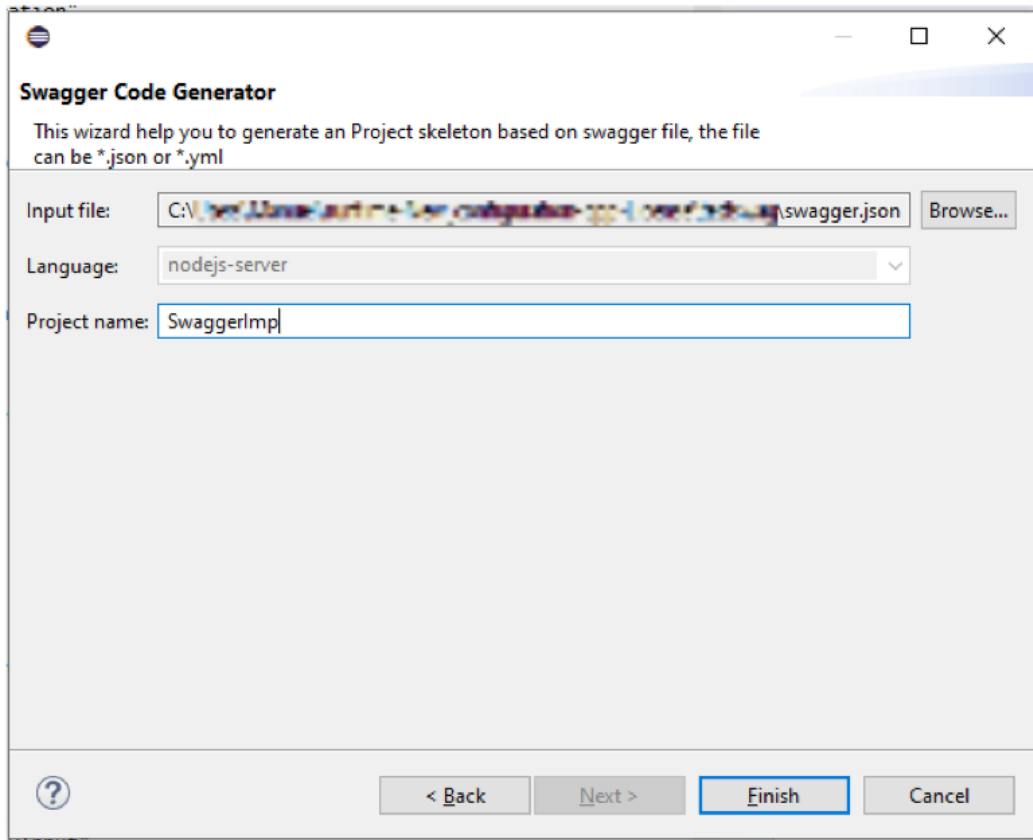
Automated generation of example data

Content types selections

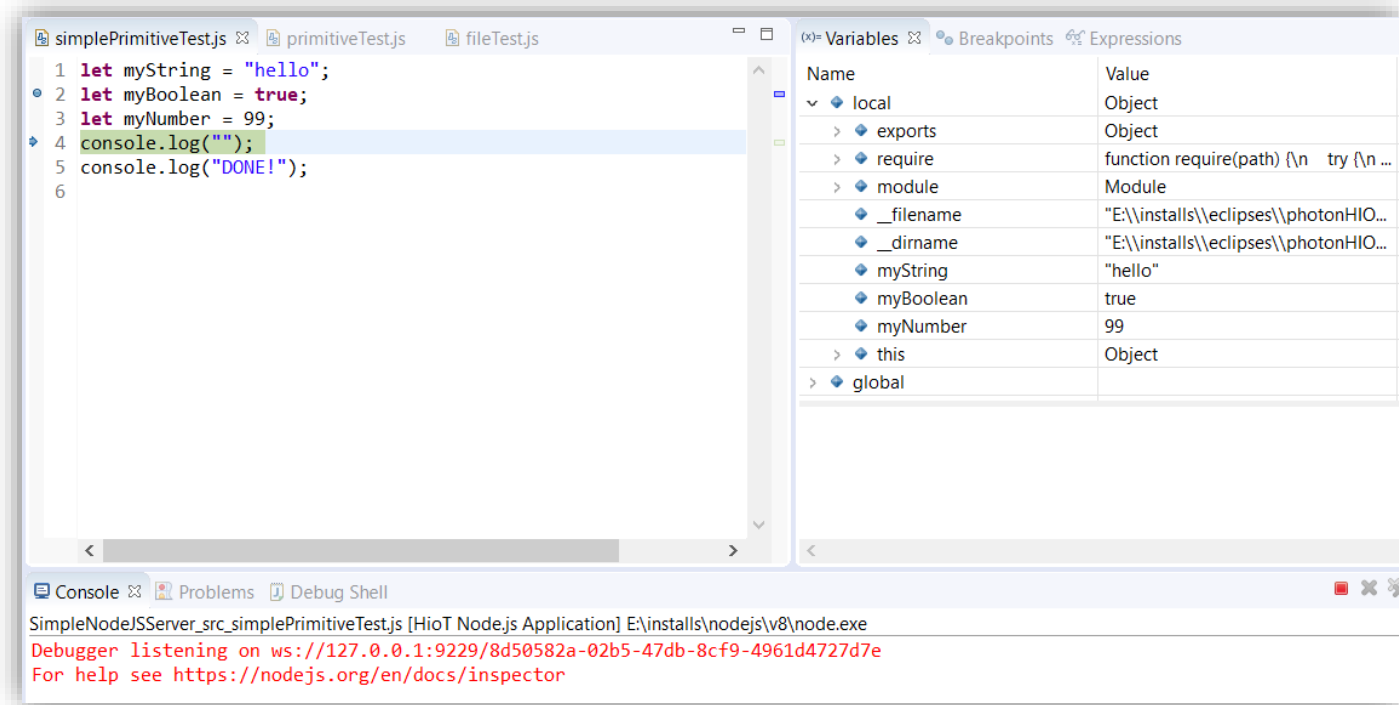
Syntax highlight and content assist



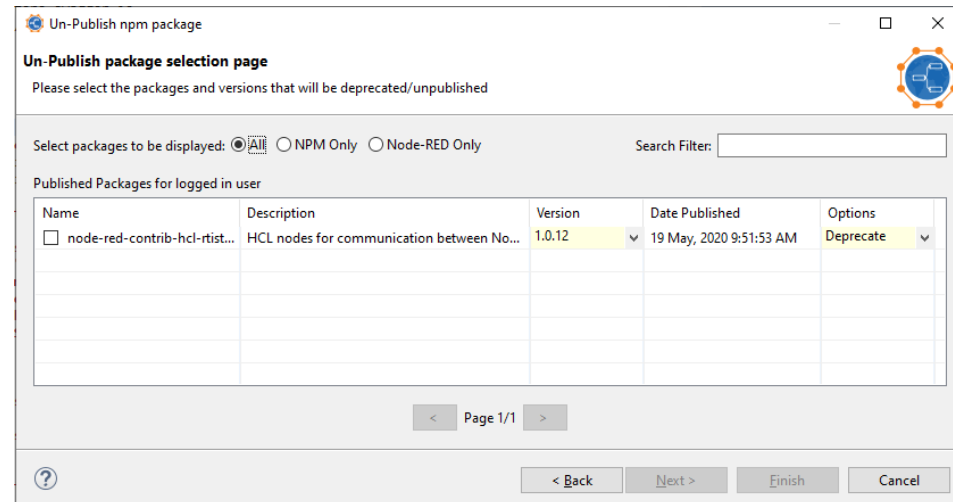
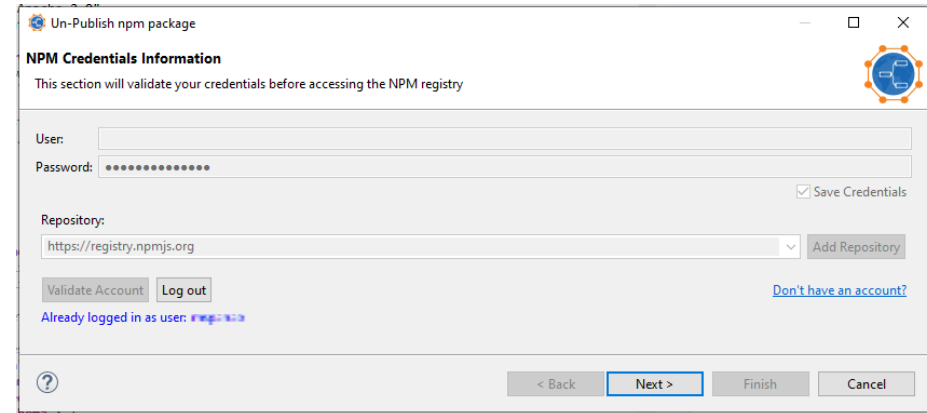
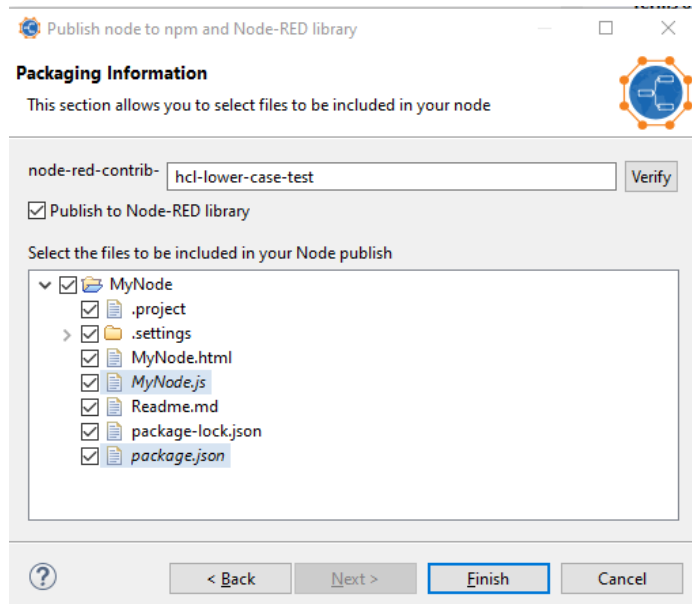
- ▶ Create development projects in different languages, such as Java, JavaScript, NodeJS, Python, etc. from a swagger specification with the Swagger Code Generator Wizard



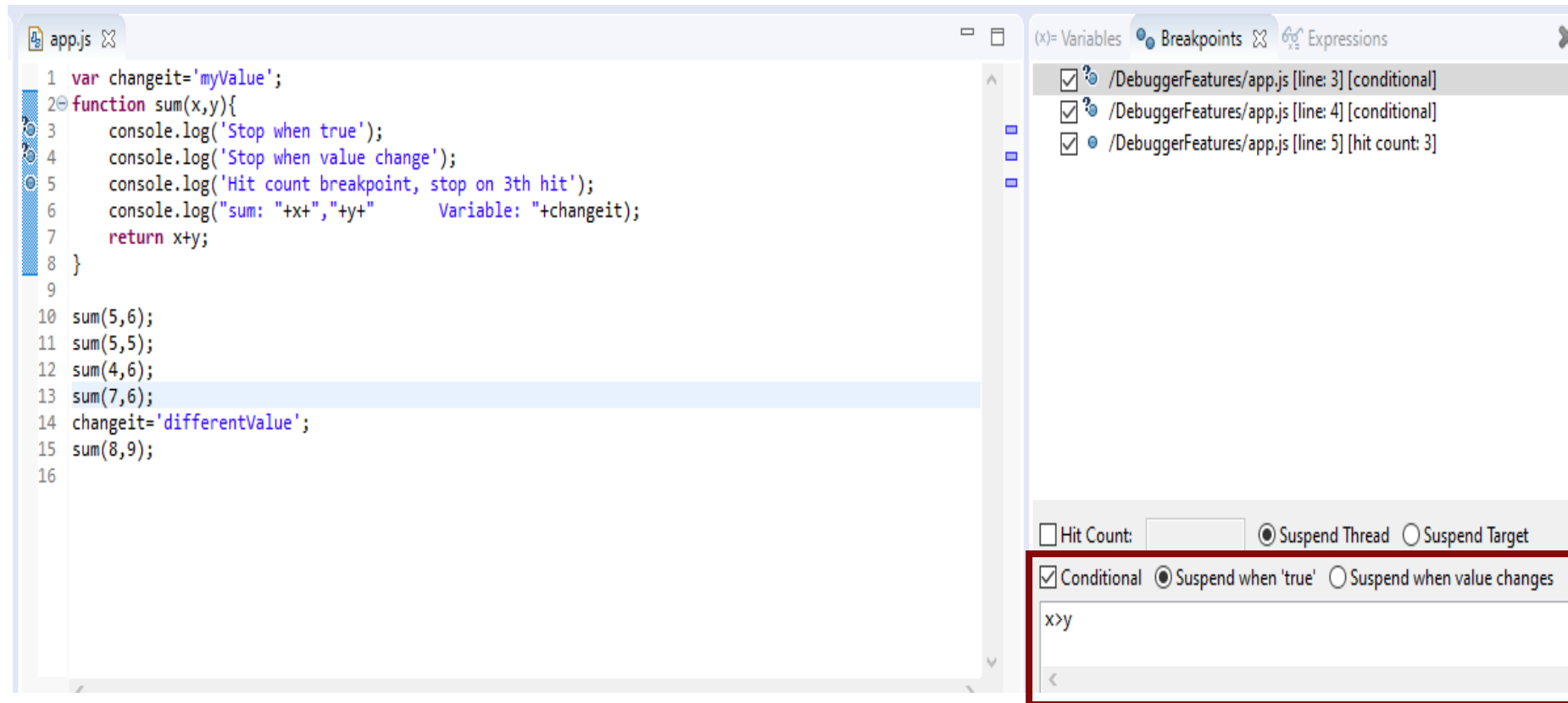
- ▶ Wizard for creating Node.js projects
- ▶ Create, start and stop a Node.js server from within the Eclipse workbench
- ▶ Debug Node.js applications with a modern JavaScript debugger
- ▶ New editor for developing HTML pages for your Node.js application using the Pug template engine



- ▶ Deploy NPM packages into the NPM repository
- ▶ Deprecate / Unpublish NPM packages from the NPM repository



- ▶ Support for conditional breakpoints
 - The Node.js debugger can be set to suspend when the conditional expression is true or when it changes



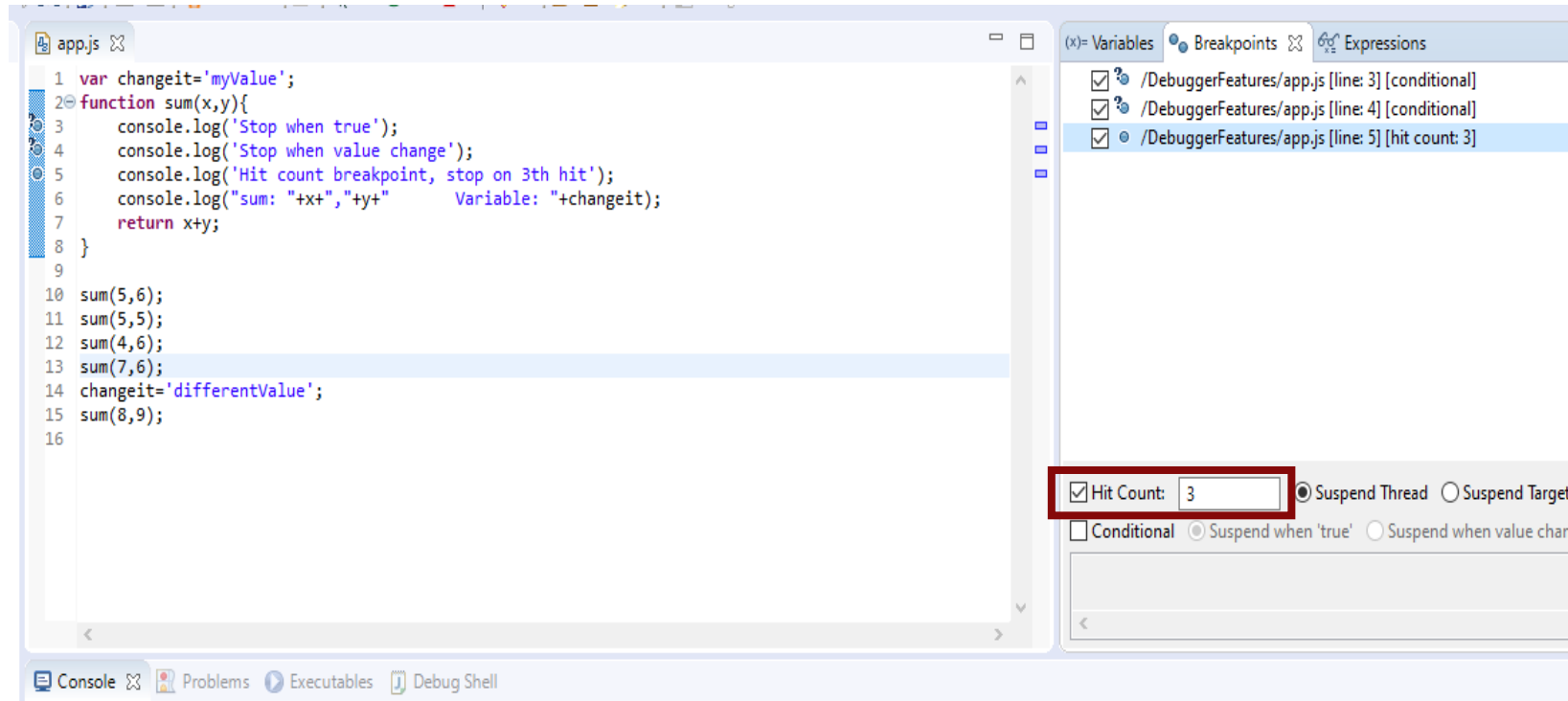
The screenshot displays the Node.js Debugger interface. On the left, the source code for `app.js` is shown with line numbers 1 through 16. The code includes a function `sum(x,y)` and several calls to `sum` and `changeit`. On the right, the Breakpoints pane shows three conditional breakpoints:

- /DebuggerFeatures/app.js [line: 3] [conditional]
- /DebuggerFeatures/app.js [line: 4] [conditional]
- /DebuggerFeatures/app.js [line: 5] [hit count: 3]

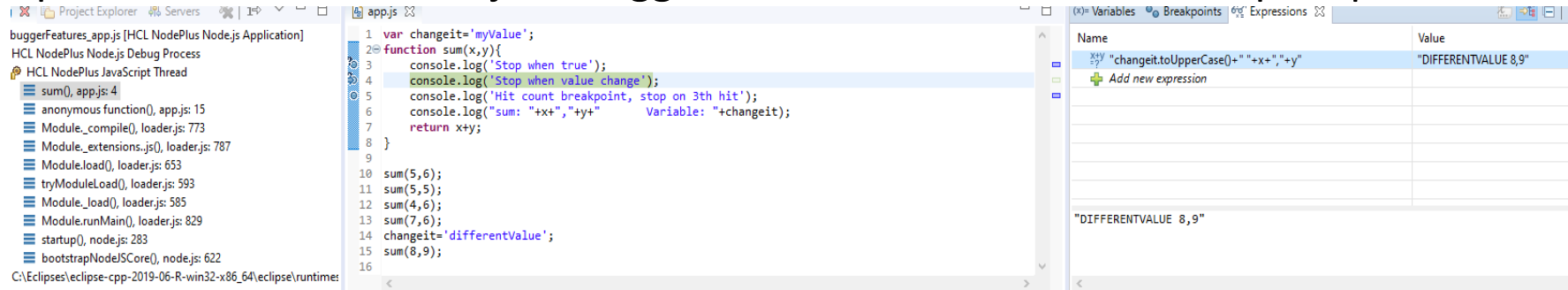
At the bottom of the Breakpoints pane, the configuration for the selected breakpoint is shown:

- Hit Count:
- Suspend Thread Suspend Target
- Conditional Suspend when 'true' Suspend when value changes
- Conditional expression: `x>y`

- ▶ Support for Hit Count breakpoints
 - The Node.js debugger can be set to suspend when a breakpoint has been hit a certain number of times



- ▶ In the Expressions tab the Node.js debugger can now evaluate a JavaScript expression in real time



The screenshot shows the Node.js debugger interface. The main editor displays a JavaScript file named `app.js` with the following code:

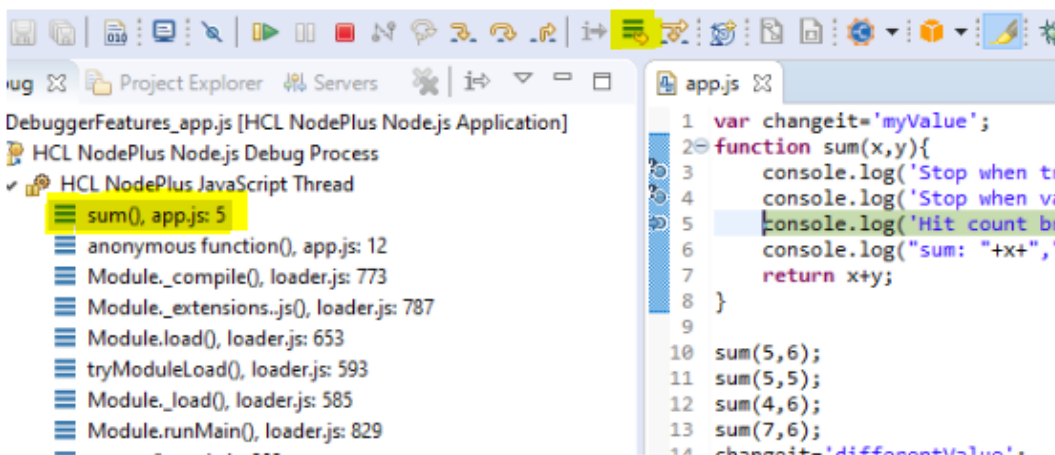
```
1 var changeit='myValue';
2 function sum(x,y){
3   console.log('Stop when true');
4   console.log('Stop when value change');
5   console.log('Hit count breakpoint, stop on 3th hit');
6   console.log("sum: "+x+", "+y+" Variable: "+changeit);
7   return x+y;
8 }
9
10 sum(5,6);
11 sum(5,5);
12 sum(4,6);
13 sum(7,6);
14 changeit='differentValue';
15 sum(8,9);
16
```

The Expressions tab on the right shows a table with the following content:

Name	Value
<code>changeit.toUpperCase()+"x"+"y"</code>	"DIFFERENTVALUE 8,9"
Add new expression	
"DIFFERENTVALUE 8,9"	

- ▶ Support for “Drop to Frame”

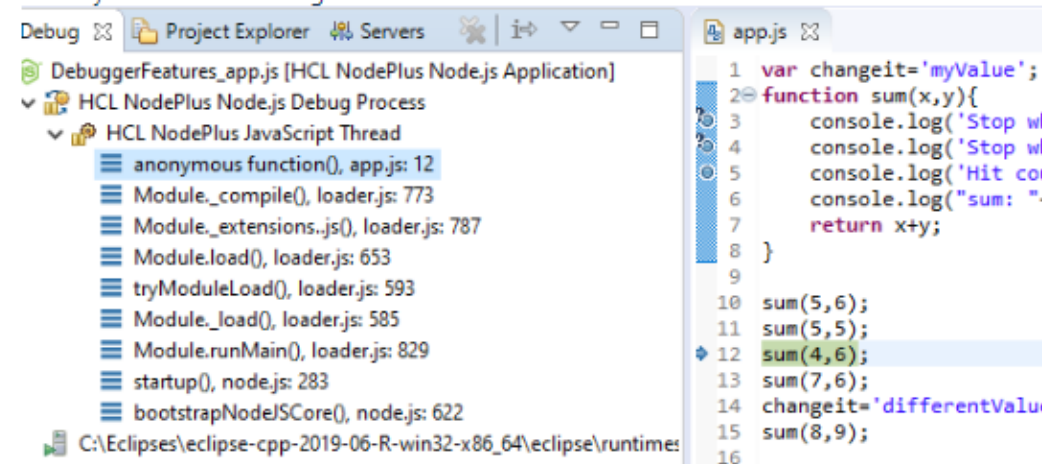
- When the application is suspended you can select a stack frame and press the “Drop to Frame” button to restart execution from there



The screenshot shows the Node.js debugger interface. The stack trace on the left shows the following frames:

- sum(), app.js: 5
- anonymous function(), app.js: 12
- Module_compile(), loader.js: 773
- Module_extensions.js(), loader.js: 787
- Module.load(), loader.js: 653
- tryModuleLoad(), loader.js: 593
- Module_load(), loader.js: 585
- Module.runMain(), loader.js: 829

The main editor displays the same JavaScript code as in the previous screenshot. The `Drop to Frame` button is highlighted in the stack trace.



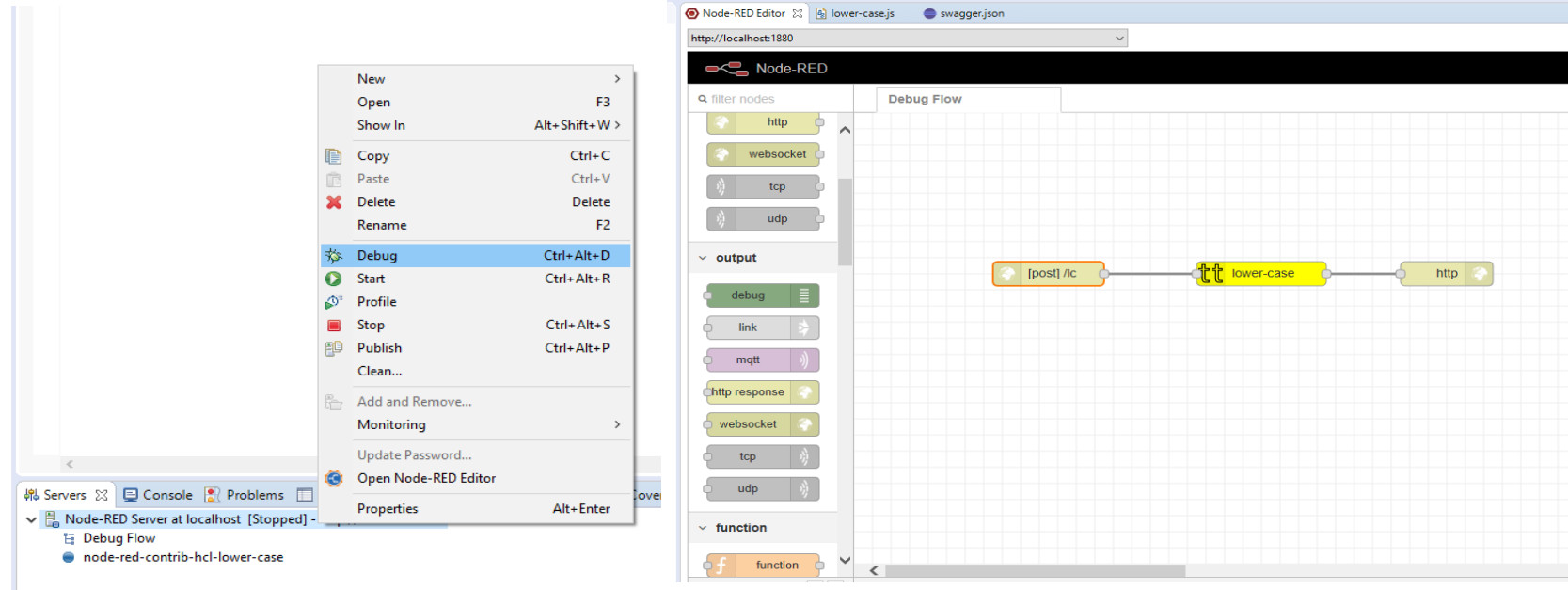
The screenshot shows the Node.js debugger interface. The stack trace on the left shows the following frames:

- anonymous function(), app.js: 12
- Module_compile(), loader.js: 773
- Module_extensions.js(), loader.js: 787
- Module.load(), loader.js: 653
- tryModuleLoad(), loader.js: 593
- Module_load(), loader.js: 585
- Module.runMain(), loader.js: 829
- startup(), node.js: 283
- bootstrapNodeJSCore(), node.js: 622

The main editor displays the same JavaScript code as in the previous screenshot. The `Drop to Frame` button is highlighted in the stack trace.

Node-RED Debugger

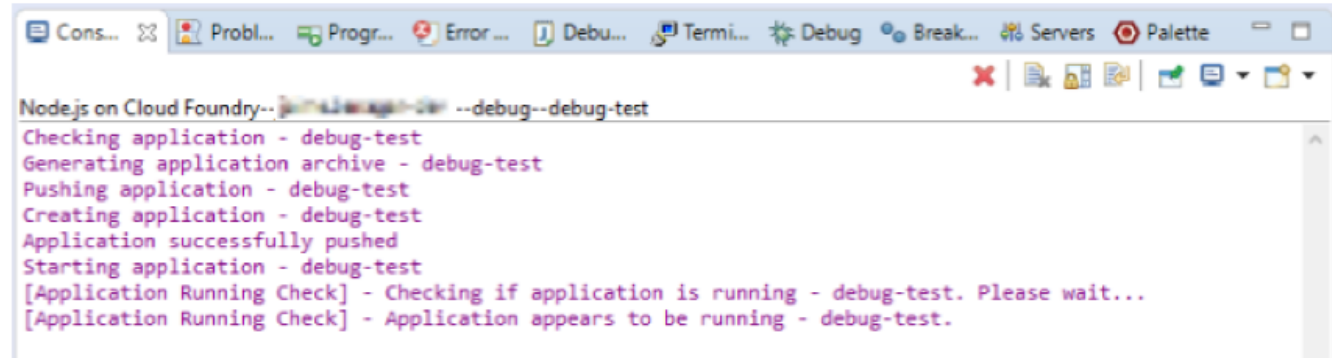
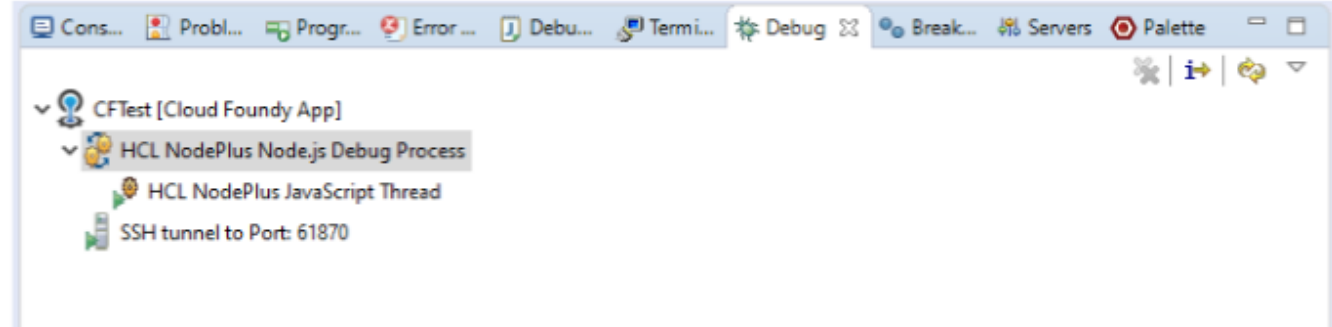
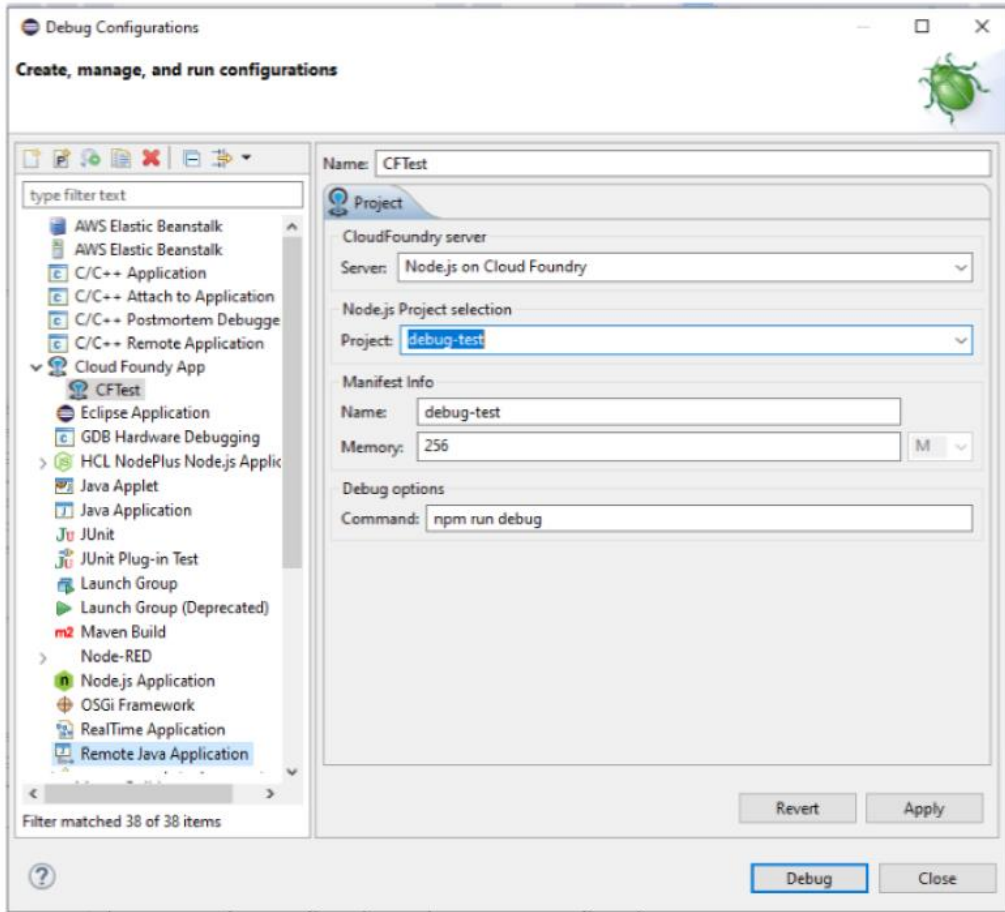
- ▶ You can use NodePlus to debug nodes deployed into Node-RED by enabling 'run in debug' mode



Debug Node.js Applications on Cloud Foundry



- ▶ You can now debug Node.js applications that are running on Cloud Foundry



HCL

*Relationship*TM
BEYOND THE CONTRACT

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

 WATCH THE FILM