Debugging NaCl module running in Emulator

Published 2014-10-28 | (Compatible with SDK 5.0,5.1 and 2014 models)

> Tutorial illustrating how to enable debugging of Native Client (NaCl) module running in Emulator.

Contents

Introduction

This tutorial presents the way to enable debugging of NaCl modules running inside Emulator and using SmartTV IDE's visual debugger.

Up to now debugging of NaCl modules was only possible using Chrome browser (see Run the Project section in Smart TV SDK PNaCl IDE Tutorial). Starting from version 5.0 of the SmartTV SDK debugging of NaCl modules running inside Emulator is also possible. This has some advantages:

SmartTV application doesn't need to be modified as it often was for running inside Chrome (Chrome doesn't support SmartTV frameworks for example)

Emulator supports almost all of SmartTV APIs, so more functionality of the application can be tested.

Technical background

The NaCl runtime already has full featured debugging mode, the so called debug stub [1]. Chrome browser already takes advantage of this feature, but now it is also available in the SmartTV Emulator. The debug stub acts as a GDB [2] server and allows for connecting GDB clients using Remote Serial Protocol (RSP) [3]. This setup covers typical debugging needs including stepping through the code, setting breakpoints and viewing values of variables.

When debug stub is enabled it stops NaCl module execution. Then it waits for a remote GDB client connection on port 4014. After the connection is established module's execution can be continued and debugging functions can be invoked from the GDB client.

The NaCl SDK provides a modified GDB client (**nacl-gdb**) which is internally used by the SmartTV SDK IDE. The IDE, when configured properly, allows for connecting to the debug stub in Emulator and to perform debugging using visual debugger in the Integrated Development Environment (IDE).

Configuring VirtualBox

By default the VirtualBox machine hosting Emulator doesn't allow incoming network connections. This blocks connecting with the GDB client running on the host OS to the debug stub running inside VirtualBox.

To change this the port forwarding feature of the VirtualBox network adapter can be used. Following settings will allow for the packets send to host at port 4014 to be forwarded to the virtual machine one the same port.

1. Start VirtualBox Manager.
2. Select Emulator's virtual machine.
3. Open Settings.
4. Go to Network section.
5. Select Adapter 1.
6. Expand Advanced section.
7. Click Port Forwarding button.
8. Add new rule and set both Host Port and Guest Port to 4014. See the following image for reference.

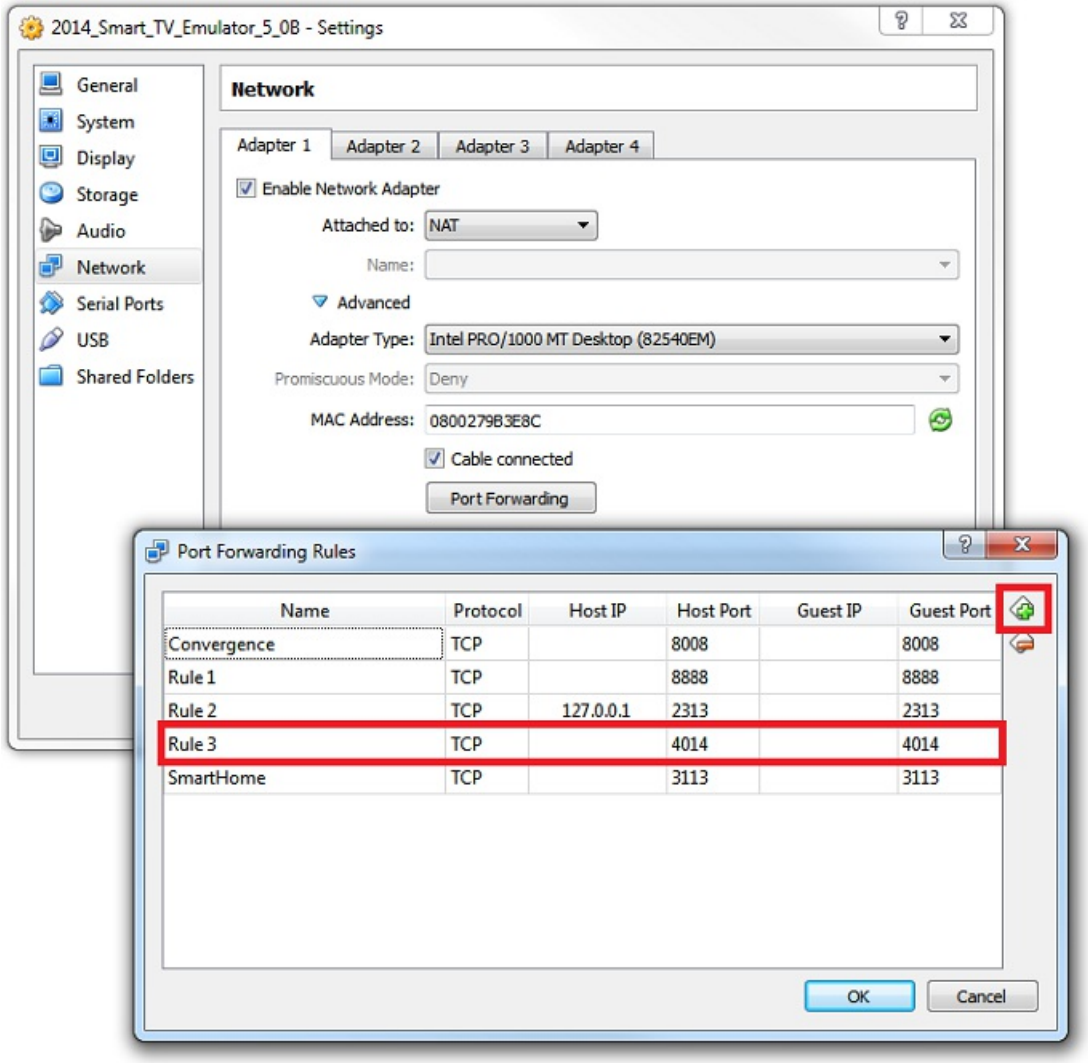

Figure 1. Setting port forwarding rule.

> Warning
>
> Make sure than only one VirtualBox instance with this port forwarding rule is opened at the same time. Also, no Chrome browser instances with enabled NaCl debugging can be running. Third party software listening on port 4014 needs to be disabled as well.

Enabling debug stub

The NaCl runtime needs to be informed if the debug stub is to be enabled or not. By default it is disabled to allow NaCl modules to execute uninterrupted.

Current solution for enabling the stub involves adding a debug attribute inside the HTML embed or object tag used for embedding NaCl module in the application. Add or remove the debug tag to respectively enable or disable debugging for particular NaCl module.

After the debug attribute is added execution of NaCl module won't continue until GDB client is connected.

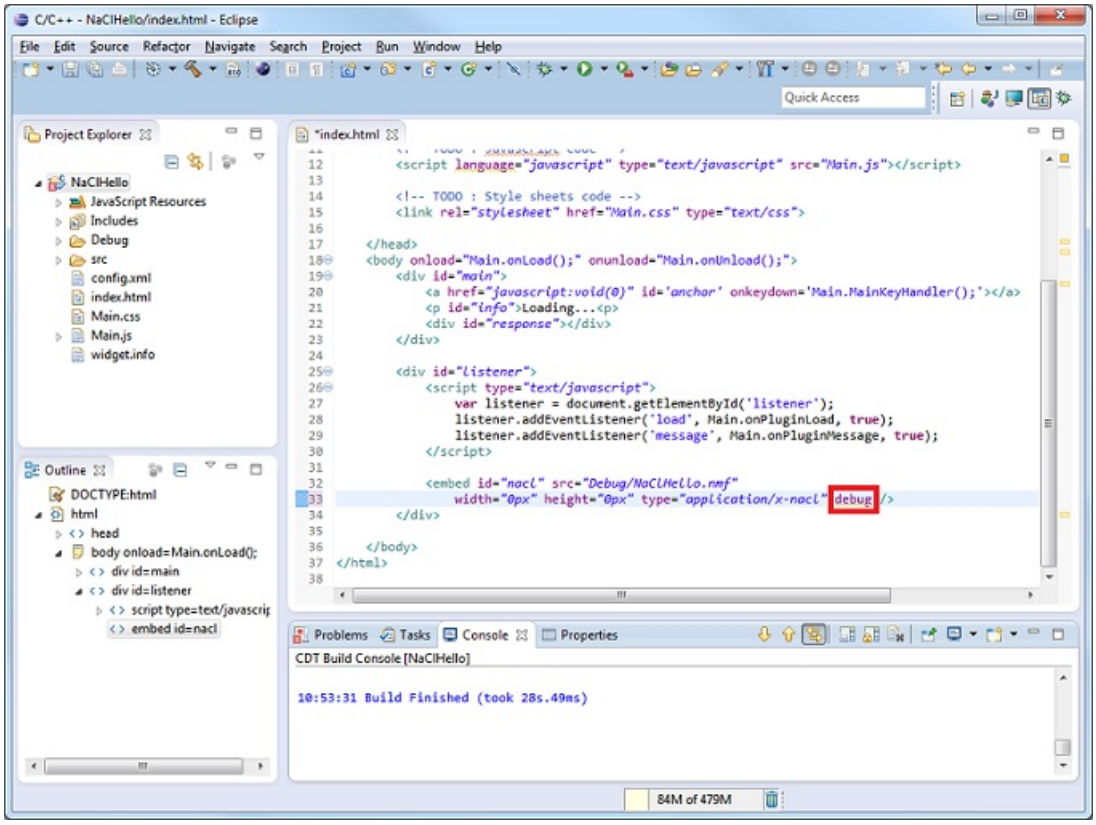

Figure 2. The debug attribute added to enable NaCl debug stub.

Preparing project in IDE

If no existing NaCl project is available, **this simple one** can be used to check the debugging features. Please import it into IDE's workspace before continuing.

1. Select File ▸ Import... from the main menu.
2. In the Import window choose General ▸ Existing Projects into Workspace.
3. Point the Select root directory field to the folder where the above mentioned project was extracted.
4. Project NaClHello should now be present in the Projects list.
5. Check the Copy projects into workspace check box.
6. Click Finish

Compiling project in debugging mode

For the debugging to be possible the project needs to be compiled in debug mode. This causes the debug symbols and source file paths to be embedded in the resulting binary. With the assumptions that the project was created with the default settings (see Smart TV SDK PNaCl IDE Tutorial) following actions will ensure the project is compiled in debug mode.

1. Click project name in the Project Explorer tree to select it.
2. In the IDE's main menu select Project ▸ Build Configurations ▸ Set Active ▸ 1 Debug

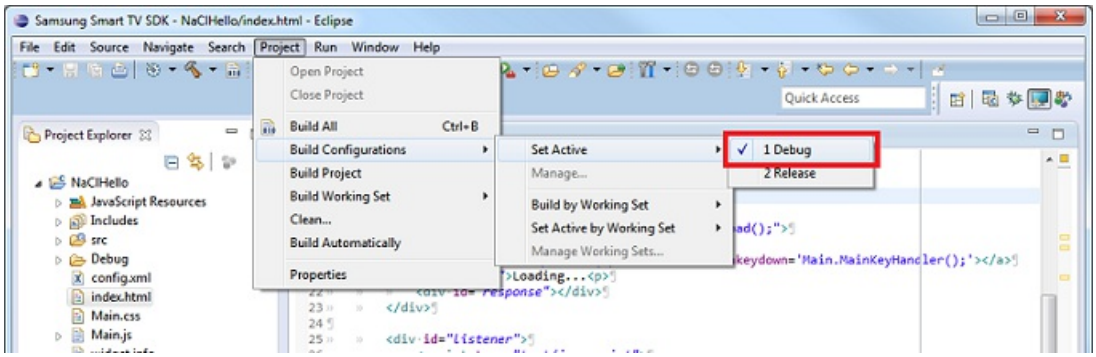

Figure 3. Setting debug build configuration.

3. Select Project ▸ Build Project

Creating debug run configuration

Debug configuration is used to run debug the compiled NaCl module. Among all it points to the NaCl module to be debugged, to the GDB executable to be used and sets the remote IP address and port to connect to. Follow these steps to create it.

1. Click project name in the Project Explorer tree to select it.
2. In the IDE's main menu select Run ▸ Debug Configurations...
3. Double click C++ Samsung PNaCl Debug.

The debug configuration should be created.

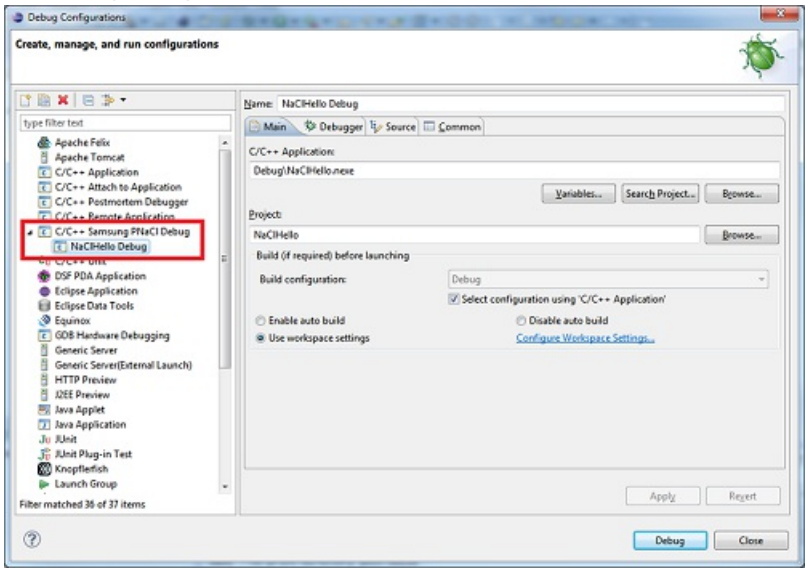

Figure 4. Newly created debug configuration.

> Important
>
> For PNaCl features to work properly your environment needs to be set correctly. Please follow the Prerequisites section in Smart TV SDK PNaCl IDE Tutorial carefully.

Debugging

Starting debugging session

1. Make sure the debug attribute is added in the HTML file (as described above in Enabling debug stub)
2. Open the application in Emulator. At this moment the NaCl module execution will be stopped.

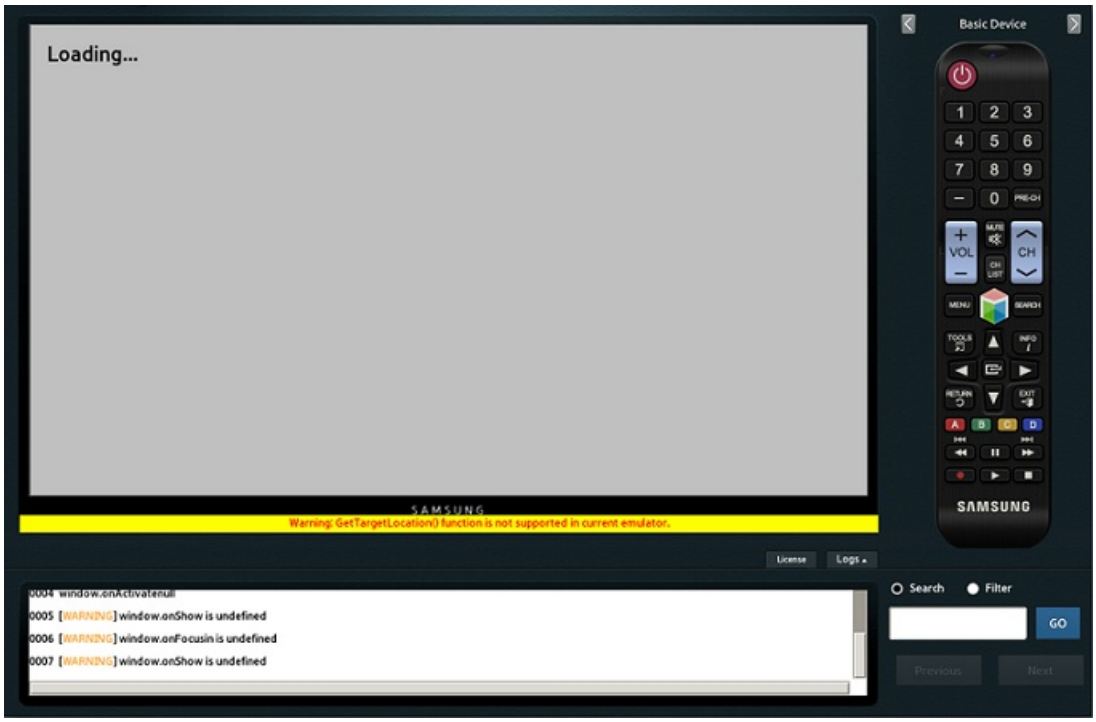

Figure 5. Example NaCl application module stopped after debug stub was enabled.

3. Run the debug configuration. Select Run ▸ Debug Configurations... from main menu. Select configuration created above in Creating debug run configuration. Then click Debug button.

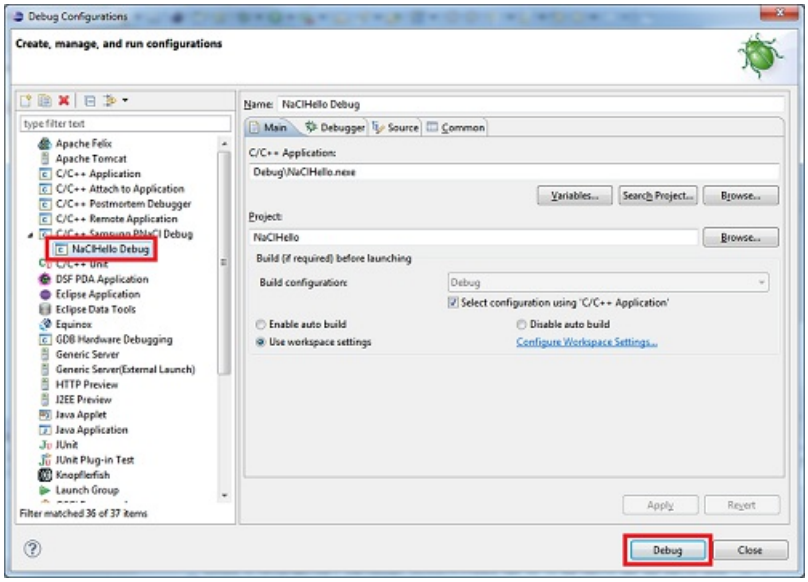

Figure 6. Selecting debug configuration.

4. The IDE will want you to switch to the Debug perspective. There all the debug features are available.

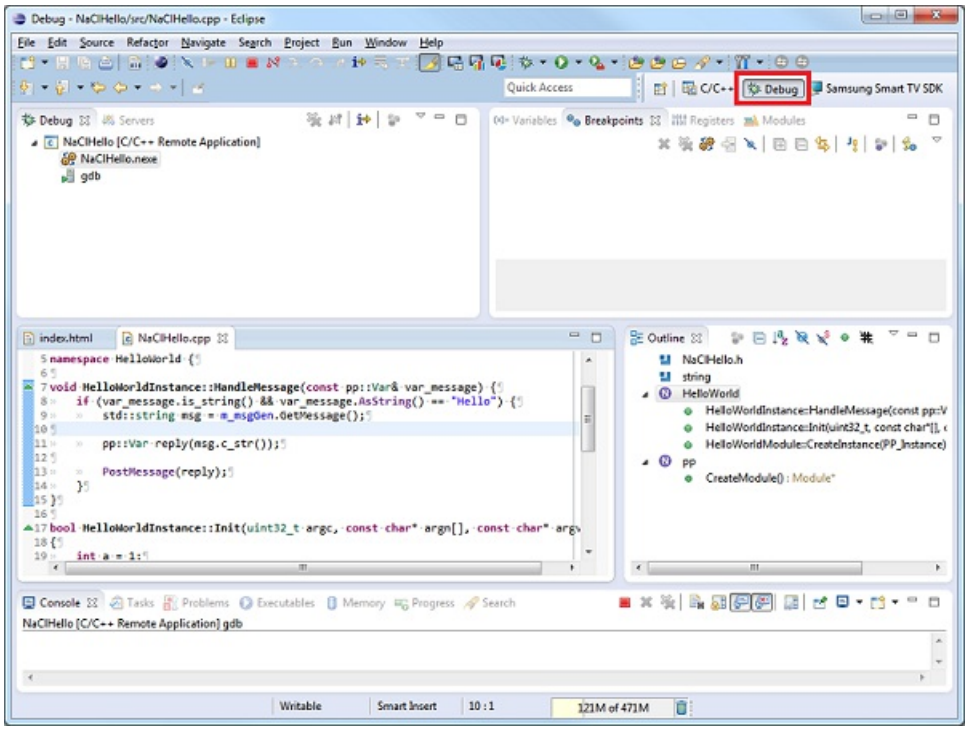

Figure 7. Debug perspective opened after running debug configuration.

Setting breakpoints

To set a breakpoint simply double click the left side margin at a line where you want to break code execution.

Figure 8. Setting breakpoint.

The list of all created breakpoints is available from the Debug perspective by going to Window ▸ Show View ▸ Breakpoints. There breakpoints can be disabled, enabled or removed.

If the code execution path will reach the breakpoint, NaCl module will be paused.

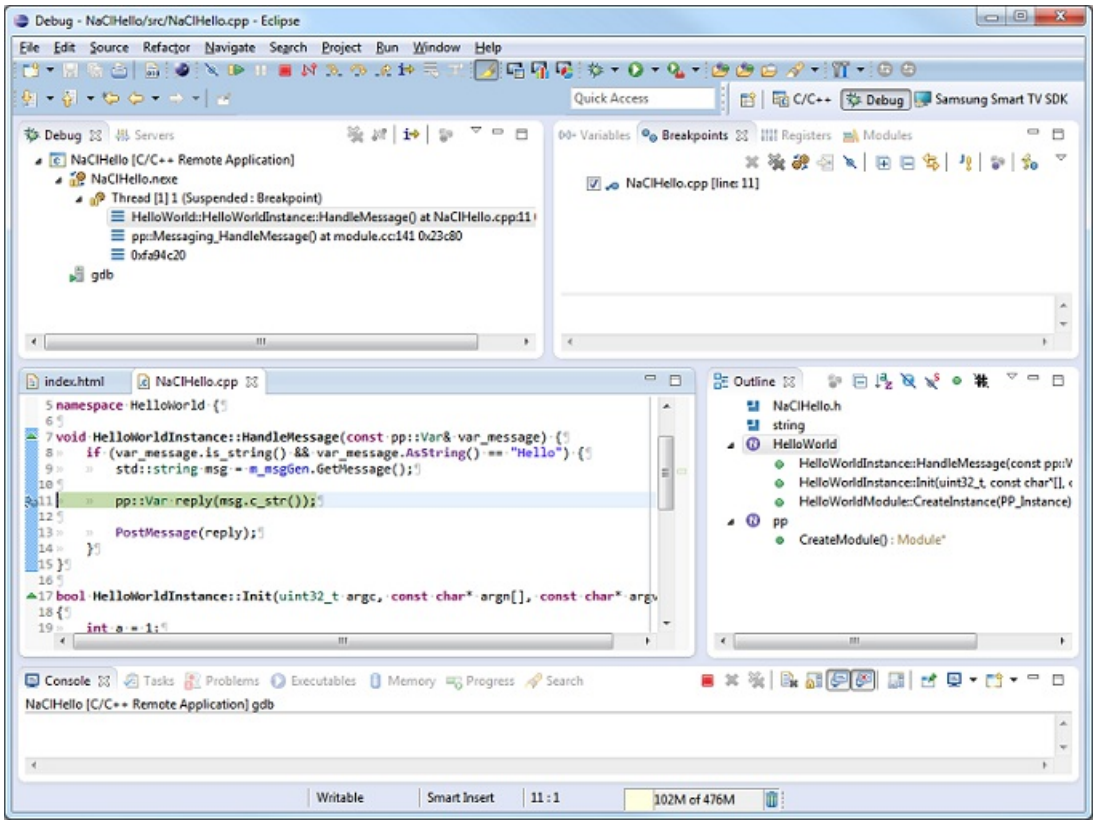

Figure 9. Code execution stopped at a breakpoint.

Stepping through code

When code execution gets paused due to a breakpoint being reached it can be manually controlled using following keyboard shortcuts.

F5 (Step Into) - Steps into a function if there is one in current line.

F6 (Step Over) - Executes current code line and moves to next

F7 (Step Return) - Goes out of current function to the previous function in call hierarchy

> Note
>
> In case the above keyboard shortcuts seem to have no effect make sure a suspended thread is selected for debugging. The Debug view in Debug perspective lists all the threads. Click a thread in the view to select it for debugging. The keyboard shortcuts should be available again.

Viewing variables

After the code execution was paused the current value's of variables within current scope can be viewed. The simplest way to do it is to hover mouse cursor over the variable in the source code.

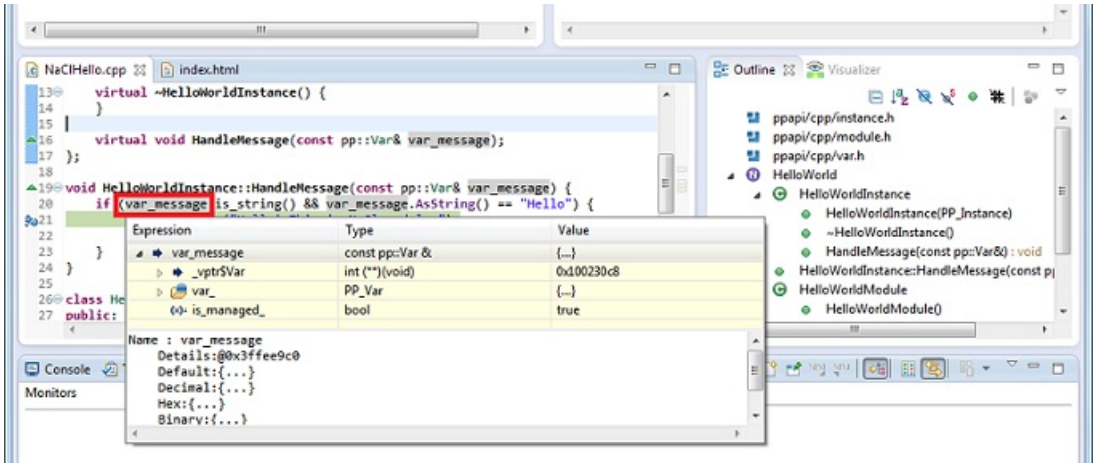

Figure 10. Viewing variable's value by mouse hovering.

Footnotes

[1]	Getting started with debug stub, **http://www.chromium.org/nativeclient/how-tos/debugging-documentation/debugging-with-debug-stub-recommended/getting-started-with-debug-stub**

[2]	GDB: The GNU Project Debugger, **http://www.sourceware.org/gdb**

[3]	Remote Serial Protocol, **https://sourceware.org/gdb/onlinedocs/gdb/Remote-Protocol.html**