CDT Testing 8.6.0 - Summary

CDT Version:	8.6 RC2						
GDB/gdbserver version							
Date Started:							
Date Completed:							
Section	**Content**	**To do**	**Pass**	**Fail**	**Started**	**Total**	
1	Installation	0	20	0	0	20	
2	Codan	0	3	0	0	3	
3	Debug	0	186	3	0	189	With comments
	Total:	**0**	**189**	**3**	**0**	**192**	
		Open	**Fixed**	**Total**		**Regressions**	
	Bug Reports	0	0	0		0	

CDT Testing 8.6.0 - Installation

	Section	Pass	Fail	To do	Started	Comment
	Installation	20	0	0	0	0
	Note: The information about the EPP and update sites to use are usually posted on epp-dev mailing list					

Step	Test Case	Action	Verification		Tester	Comment
1	**Verify C/C++ EPP Package RC1**					
1.1	Download EPP Package	Download, extract and start EPP package	EPP Package starts		Not done this time	
1.2	Version of CDT Features	Go to Help -> About Eclipse -> Installion Details	Verify that the CDT features and plug-ins are present and have the correct version		Not done this time	
1.3	C/C++ Perspective	Open C/C++ perspective	C/C++ perspective opens		Not done this time	
1.4	Compilation and code analysis	Create a simple Hello world program using the project wizard.	Verify that the program compiles and that there are no errors (code analysis)		Not done this time	
1.5	Debug	Debug the program (right-click Debug As, Local C/C++ Application)	Verify that the debug session starts, stepping is possible and terminate works.		Not done this time	
2	**Verify C/C++ EPP Package RC2**					
2.1	Download EPP Package	Download, extract and start EPP package	EPP Package starts	Pass	Marc-Andre Laperle	
2.2	Version of CDT Features	Go to Help -> About Eclipse -> Installion Details	Verify that the CDT features and plug-ins are present and have the correct version	Pass	Marc-Andre Laperle	
2.3	C/C++ Perspective	Open C/C++ perspective	C/C++ perspective opens	Pass	Marc-Andre Laperle	
2.4	Compilation and code analysis	Create a simple Hello world program using the project wizard.	Verify that the program compiles and that there are no errors (code analysis)	Pass	Marc-Andre Laperle	
2.5	Debug	Debug the program (right-click Debug As, Local C/C++ Application)	Verify that the debug session starts, stepping is possible and terminate works.	Pass	Marc-Andre Laperle	
3	**Verify C/C++ EPP Package RC3**					
3.1	Download EPP Package	Download, extract and start EPP package	EPP Package starts	Pass	Marc-Andre Laperle	
3.2	Version of CDT Features	Go to Help -> About Eclipse -> Installion Details	Verify that the CDT features and plug-ins are present and have the correct version	Pass	Marc-Andre Laperle	
3.3	C/C++ Perspective	Open C/C++ perspective	C/C++ perspective opens	Pass	Marc-Andre Laperle	
3.4	Compilation and code analysis	Create a simple Hello world program using the project wizard.	Verify that the program compiles and that there are no errors (code analysis)	Pass	Marc-Andre Laperle	
3.5	Debug	Debug the program (right-click Debug As, Local C/C++ Application)	Verify that the debug session starts, stepping is possible and terminate works.	Pass	Marc-Andre Laperle	
4	**Verify C/C++ EPP Package RC4**					
4.1	Download EPP Package	Download, extract and start EPP package	EPP Package starts	Pass	Marc Khouzam	
4.2	Version of CDT Features	Go to Help -> About Eclipse -> Installion Details	Verify that the CDT features and plug-ins are present and have the correct version	Pass	Marc Khouzam	
4.3	C/C++ Perspective	Open C/C++ perspective	C/C++ perspective opens	Pass	Marc Khouzam	
4.4	Compilation and code analysis	Create a simple Hello world program using the project wizard.	Verify that the program compiles and that there are no errors (code analysis)	Pass	Marc Khouzam	
4.5	Debug	Debug the program (right-click Debug As, Local C/C++ Application)	Verify that the debug session starts, stepping is possible and terminate works.	Pass	Marc Khouzam	
5	**Verify Update Site**					
5.1	Luna Update Site	Download Eclipse standard and install all CDT features from main Luna SR2 testing Update site http://download.eclipse.org/releases/maintenance	Verify that installation was successful	Pass	Marc Khouzam	
5.2	CDT Update Site	Download Eclipse standard and install all CDT features from the CDT Update site http://download.eclipse.org/tools/cdt/builds/luna/milestones	Verify that installation was successful	Pass	Marc Khouzam	

5.3	Upgrade using Luna Update Site	Download Eclipse standard from Luna SR1 and install all CDT features from main Luna Update site. http://download.eclipse.org/releases/luna Try to update the installation using the testing Luna SR2 update site. http://download.eclipse.org/releases/maintenance	Verify that installation was successful	Pass	Marc Khouzam	
5.4	Upgrade using CDT Update Site	Download Eclipse standard from Luna SR1 and install all CDT features from the Luna SR1 CDT Update site. http://download.eclipse.org/tools/cdt/releases/8.5 Try to update the installation using the Luna SR2 CDT Update site. http://download.eclipse.org/tools/cdt/builds/luna/milestones	Verify that installation was successful	Pass	Marc Khouzam	
5.5	Upgrade from previous EPP	Download Eclipse previous C/C++ EPP package. Try to upgrade using both update sites: 1) https://hudson.eclipse.org/packaging/job/luna.epp-tycho-build/128/artifact/org.eclipse.epp.packages/archive/repository/ 2) http://download.eclipse.org/releases/staging/ or http://download.eclipse.org/releases/maintenance/ (for a SR release) The information about the update sites to use is usually posted on epp-dev	Verify that installation was successful	Pass	Marc Khouzam	

	Section	Pass	Fail	To do	Started	Comment
	Codan	3	0	0	0	0

Step	Test Case	Action	Verification		Tester	Comment
1	**Preparation**					
1.1	Step 1	Open C/C++ perspective	Perspective opens with correct views	Pass	Marc Khouzam	
2	**Tests**					
2.1	Preferences	Open Preferences->C/C++->Code Analysis page	Verify the page exists	Pass	Marc Khouzam	
2.2	Syntax check	Enable a type of error and introduce that error in the source code	Verify that the error is detected immediately when typing	Pass	Marc Khouzam	

CDT Testing 8.6.0 - Debug

	Section	Pass	Fail	To do						Comment
	Debug	186	3	0	0					8

Step	Test Case	Action	Verification	Linux	Tester	Windows	Tester	Mac	Tester	Comment
1	**Preparation**									
1.1	Step 1	Open C/C++ perspective	Perspective opens with correct views	Pass	Marc Khouzam					
2	**Local Debug**									
2.1	Perspective switch	Launch a local debug session in non-stop mode	Verify the perspective is changed to the Debug perspective	Pass	Marc Khouzam					GDB on Mac doesn't support non-stop and the debug session never terminates after trying to debug in non-stop. This might be related to https://bugs.eclipse.org/bugs/show_bug.cgi?id=427410
2.2	Debug session	Inspect Debug view	Verify there are nodes for the launch, the process, threads and stack frames, and one gdb node	Pass	Marc Khouzam					
2.3	Console selection	Select the 'gdb' node in the Debug view	Verify the gdb console appears in the console view	Pass	Marc Khouzam					
2.4	Stepping	Press the different stepping buttons	Verify stepping works as expected	Pass	Marc Khouzam					
2.5	Resume	Press the resume button while a thread is stopped	Verify resume works as expected	Pass	Marc Khouzam					
2.6	Suspend	Press the suspend button while a thread is running	Verify suspend works as expected	Pass	Marc Khouzam					
2.7	Breakpoint interrupt	While the target is running, set a breakpoint	Verify that the target is temporarily interrupted to set the breakpoint and then resumed	Pass	Marc Khouzam					
2.8	Run-to-line 1	Select a line in the current method and press Ctrl-R	Verify execution continue until that line	Pass	Marc Khouzam					
2.9	Run-to-line 2	Select a line in a different method and press Ctrl-R	Verify execution continue until that line	Pass	Marc Khouzam					
2.10	Registers per stack frame	Make sure there are more than one stack frames visible	Verify that at least the stack pointer register changes depending on the selected stack frame	Pass	Marc Khouzam					
2.11	Variables view	Look at variables view	Verify local variables are displayed for current frame	Pass	Marc Khouzam					
2.12	Variables view update	Change stack frame in debug view	Verify local variables are displayed for new frame	Pass	Marc Khouzam					
2.13	Expressions view	Create a valid expression in the expressions view	Verify expression value is shown for current frame	Pass	Marc Khouzam					
2.14	Expressions view update	Change stack frame in debug view	Verify expression value is updated (to maybe an error) for the new frame	Pass	Marc Khouzam					
2.15	Registers view	Look at Registers view	Verify registers are shown with their values	Pass	Marc Khouzam					
2.16	Memory view	Add a memory monitor	Verify the memory corresponding to the monitor is shown	Pass	Marc Khouzam					
2.17	Memory Browser view	Put an address in the address box	Verify the memory corresponding to the address is shown	Pass	Marc Khouzam					
2.18	Connect button	Press the connect button in the Debug view	Verify a dialog showing all processes of the system is displayed	Pass	Marc Khouzam					
2.19	New...	Press the New... button from the connect dialog	Verify a prompt for a binary is displayed	Pass	Marc Khouzam					
2.20	New process	Select a valid path for a binary in the prompt	Verify that the proper binary is added to the debug session	Pass	Marc Khouzam					
2.21	Cores	Look at Debug view	Verify that the 'cores' are displayed next to each process and each thread node	Pass	Marc Khouzam					
2.22	Show full path option	Toggle "Show full path" option in Debug view	Verify that the full path of both the frames and process is shown or not shown according to the option	Pass	Marc Khouzam					
2.23	Show only suspended threads	Toggle the preference "Show only suspended threads"	Verify that all running threads disappear and that a text saying how many threads are hidden is shown next to the process node	Pass	Marc Khouzam					
2.24	Show thread names in Debug view	Either debug a program that sets thread names, or expect to see the process name use as thread names	Verify that the thread names are shown in the Debug view next to each thread	Pass	Marc Khouzam					
3	**Local-attach Debug**									
3.1	Attach launch	Launch a local-attach debug session in non-stop mode	Verify a dialog showing all processes of the system is displayed	Pass	Alvaro					
3.2	Cancel attach	Press the Cancel button	Verify the entire launch is terminated without error	Pass	Alvaro					
3.3	Preparation	From the shell, start three long running processes	Processes are started	Pass	Alvaro					
3.4	Attach launch	Launch a local-attach debug session in non-stop mode	Verify a dialog showing all processes of the system are displayed	Pass	Alvaro					
3.5	Multi-select	Select multiple entries	Verify multi-selection is supported	Pass	Alvaro					
3.6	Multi-attach	Select the three processes that were started earlier	Verifies that all three process start being debugged without being interrupted	Pass	Alvaro					
3.7	Suspend	Interrupt the second process	Verify the second process is interrupted	Pass	Alvaro					
3.8	Set breakpoint	Set a breakpoint in the second process	Verify breakpoint is set	Pass	Alvaro					
3.9	Resume	Resume the second process	Verify that the second process resumes then stops at the breakpoint	Pass	Alvaro					
3.10	Breakpoint interrupt	Set a breakpoint in the first process while it is running	Verify that the first process stops at the breakpoint	Pass	Alvaro					
3.11	Memory view multi-process		Verify that memory monitors are per process	Pass	Alvaro					
3.12	Memory browser multi-process		Verify that memory browser tabs are per process	Pass	Alvaro					
3.13	Registers multi-process		Verify that the list of registers is fetched for each process of the debug session	Pass	Alvaro					
3.15	Detach running	Detach from a running process	Verify that the process keeps on running in the OS but is no longer debugged	Pass	Alvaro					
3.16	Detach suspended	Detach from a suspended process	Verify that the process starts running again in the OS but is no longer debugged	Pass	Alvaro					
3.17	Re-attach running	Re-attach to the running process that was detached	Verify the process is debugged again	Pass	Alvaro					
3.18	Re-attach suspended	Re-attach to the suspended process that was detached	Verify the process is debugged again	Pass	Alvaro					
3.19	Terminate running	Terminate from a running process	Verify that the process is terminated in the OS	Pass	Alvaro					
3.20	Terminate suspended	Terminate from a suspended process	Verify that the process is terminated in the OS	Pass	Alvaro					Does not work if the process is the last one in the session. Known issue.
3.21	State	Look at Debug view	Verify only a single process is left to debug	Pass	Alvaro					
3.22	Cancel attach	Press the connect button on the Debug view and then Cancel	Verify that the prompt disappears and that the debug session stays unchanged (one process being debugged)	Pass	Alvaro					

3.23	Cancel new	Press the connect button then New.. and then Cancel	Verify that the prompt disappears and that the debug session stays unchanged (one process being debugged)	Pass	Alvaro					
3.24	New process	Press the connect button then New.. and then select a valid binary	Verify that this binary starts being debugged	Fail	Alvaro					Only works if the existing process has an interrupted thread. Known issue. It also works if all existing processes are running but there are no break points defined
3.25	State	Look at Debug view	Verify that there are two processes being debugged	Pass	Alvaro					
3.26	Dynamic-printf	Create a dprintf within one attached program and run past it	Verify the dprintf is printed to the original process console outside of Eclipse	Pass	Alvaro					
4	**Remote-attach Debug**									
4.1	Preparation	From the shell, start 'gdbserver –multi :9999' using the latest gdbserver	gdbserver started	Pass						
4.2	Remote-attach	Launch a remote attach debug session in non-stop mode	Verify Debug view shows a new launch with only the launch node and 'gdb' nodes	Pass						
4.3	Preparation	From the shell, start three long running processes	Processes are started	Pass						
4.4	Connect button	Press the connect button on the Debug view	Verify a dialog showing all processes of the system is displayed	Pass						
4.5	Multi-attach	Select the three processes that were started earlier	Verify a prompt for a binary is displayed and that the name of the process is shown as the title	Pass						
4.6	Binary 1	Specify the proper binary	Verify a prompt for a second binary is displayed and that the name of the process is shown as the title	Pass						
4.7	Binary 2	Specify the proper binary	Verify a prompt for a third binary is displayed and that the name of the process is shown as the title	Pass						
4.8	Binary 3	Specify the proper binary	Verifies that all three process start being debugged without being interrupted	Pass						
4.9	Suspend	Interrupt the second process	Verify the second process is interrupted	Pass						
4.10	Set breakpoint	Set a breakpoint in the second process	Verify breakpoint is set	Pass						
4.11	Resume	Resume the second process	Verify that the second process resumes then stops at the breakpoint	Pass						
4.12	Breakpoint interrupt	Set a breakpoint in the first process while it is running	Verify that the first process stops at the breakpoint	Pass						
4.13	Detach running	Detach from a running process	Verify that the process keeps on running in the OS but is no longer debugged	Pass						
4.14	Detach suspended	Detach from a suspended process	Verify that the process starts running again in the OS but is no longer debugged	Fail						got a segfault in "long1" after detaching. Might have been a fluke - seems to work now
4.15	Re-attach running	Re-attach to the running process that was detached	Verify the process is debugged again	Pass						
4.16	Re-attach suspended	Re-attach to the suspended process that was detached	Verify the process is debugged again	Pass						
4.17	Terminate running	Terminate from a running process	Verify that the process is terminated in the OS	Pass						
4.18	Terminate suspended	Terminate from a suspended process	Verify that the process is terminated in the OS	Pass						
4.19	State	Look at Debug view	Verify only a single process is left to debug	Pass						
4.20	Start new process	Press the connect button on the Debug view	Verify the "New..." button is enabled	Pass						
4.21	Start new process	Press the "New..." button in the attach dialog	Verify a new dialog pops up asking for two binary locations and arguments	Pass						
4.22	Start new process	Specify the local and remote binaries and some arguments	Verify the process is started with the specified arguments	Pass						
4.23	Dynamic-printf	Create a dprintf within one program and run past it	Verify the dprintf is printed to the original process console outside of Eclipse	Pass						
5	**Automatic Remote Debug**									
5.1	Auto-remote	Launch an automatic remote debug session	Verify the process is being debugged	Pass						
5.2	Dynamic-printf	Set a dynamic-printf and run past it	Verify the dprintf is printed to the remote console in Eclipse	Pass						
6	**Manual Remote Debug**									
6.1	Preparation	From the shell, start 'gdbserver :9999 <binaryPath>' using the latest gdbserver	gdbserver started	Pass						
6.2	Remote-attach	Launch a manual remote debug session	Verify there are nodes for the launch, the process, threads and stack frames, and one gdb node	Pass						
6.3	Dynamic-printf	Set a dynamic-printf and run past it	Verify the dprintf is printed to the console where gdbserver was started outside of Eclipse	Pass						
7	**Post-mortem Core file**									
7.1	Preparation	Start a local debug session	Debug session started	Pass	Alvaro					
7.2	Preparation	Step or resume to another method than main	Debug session interrupted in another method	Pass	Alvaro					
7.3	Preparation	Interrupt all threads	Make sure all threads are interrupted	Pass	Alvaro					
7.4	Preparation	In the gdb console type 'gcore /tmp/gcore1' to generate a core file	Verify /tmp/gcore1 is created	Pass	Alvaro					
7.5	Post-mortem launch	Start a post-mortem debug session using /tmp/gcore1	Verify the debug view shows the program stopped where the core file was generated	Pass	Alvaro					
7.6	Debug view buttons	Look at Debug view	Verify all step and resume buttons are grayed out	Pass	Alvaro					
7.7	Variables view	Look at variables view	Verify variables are shown in variables view	Pass	Alvaro					
7.8	Empty core field	Start a post-mortem debug session leaving the core file field empty	Verify a prompt for a core file is displayed	Pass	Alvaro					
7.9	Select core file	select /tmp/gcore1	Verify the core file starts being 'debugged' as it was in the previous attempt	Pass	Alvaro					
7.10	Directory in core field	Start a post-mortem debug session putting /tmp in the core file field	Verify that a prompt for a core file is displayed starting in /tmp	Fail	Alvaro					if the field is left empty, then browsing to /tmp provides no files to select from
7.11	Select core file	select /tmp/gcore1	Verify the core file starts being 'debugged' as it was in the previous attempt	Pass	Alvaro					
7.12	Cancel launch	Start a post-mortem debug session leaving the core file field empty and press cancel at the prompt	Verify the launch is terminated cleanly	Pass	Alvaro					
8	**Pretty-printing tests**									

8.1	Preparation	Hide variables and expressions view	Variables and Expressions view are not visible to avoid showing un-initialized STL structures, which could hang GDB.	Pass						
8.2	Preparation	Launch a local debug session in non-stop mode with code using Maps/Lists/Vectors	Debug session started	Pass						
8.3	Preparation	Execute until all STL variables are initialized	Execution stopped after STL vars initialized	Pass						
8.4	Pretty-printed variables view	Look at variables view	Verify that the STL structures are displayed pretty-printed in the variables view, both in the view and in the detail pane	Pass						
8.5	Pretty-printed expressions view	Look at expressions view	Verify that the STL structures are displayed pretty-printed in the expressions view, both in the view and in the detail pane	Pass						
8.6	Edit	Change the value of an STL content	Verify that the value changes as expected	Pass						
9	**Tracepoint tests**									
9.1	Preparation	Start an automatic remote debug session using non-stop	Debug session started	Pass						
9.2	Tracepoints	Create two tracepoints	Tracepoints created	Pass						
9.3	Tracepoint commands	Add the following actions to the first tracepoint: 'collect $locals' and 'collect $reg'	Verify commands sent to GDB	Pass						
9.4	Tracepoint commands 2	Add the following actions to the second tracepoint: 'collect $trace_timestamp' and 'collect <single local var>'	Verify commands sent to GDB	Pass						
9.5	Start trace experiment	Start trace execution and resume execution of program	Trace records produced in Trace control view	Pass						
9.6	Stop trace experiment	Stop trace experiment	Verify trace experiment is shown as stopped	Pass						
9.7	Trace navigation	In the Trace Control view, press the Next Record button and navigate through the collected records	Verify Debug view updates to follow the different trace records	Pass						
9.8	Variables view	Look at Variables view and Debug view	Verify that the collected data is properly displayed	Pass						
9.9	Unavailable data	Look at Variables view for data not collected	Make sure that unavailable data shows "<unavailable>"	Pass						seems that the registers are captures on both tracepoints
9.10	Stop visualization	From the Trace Control view press the Exit Visualization button	Verify the Debug view goes back to the program execution display	Pass						
9.11	Trace navigation	In the Trace Control view, press the Next Record button and navigate through the collected records	Verify Debug view updates to follow the different trace records	Pass						
9.12	Save trace data	From the Trace Control view menu, save the trace data to /tmp/tracedata	Verify /tmp/tracedata is created	Pass						
9.13	Post-mortem launch	Start a post-mortem debug session using the generated trace file from previous test case	Verify the debug view shows the program stopped where the first trace record was collected	Pass						
9.14	Debug view buttons	Look at Debug view	Verify all step and resume buttons are grayed out	Pass						
9.15	Variables view	Look at variables view	Verify variables are shown in variables view	Pass						
9.16	Unavailable data	Look at Variables view for data not collected	Make sure that unavailable data shows "<unavailable>"	Pass						
10	**Fast Tracepoint tests**									
10.1	Fast tracepoint option	Start an automatic remote debug session using non-stop with the "Fast Tracepoint" option	Debug session started	Pass	Marc Khouzam					
10.2	Fast Tracepoint	Create a tracepoint that can be set as a fast one (a fast tp needs a 4-byte instruction on a 32-bit machine, and a 5-byte one on a 64-bit machine)	Verify that a fast tracepoint is created	Pass	Marc Khouzam					
10.3	Normal tracepoint	Create a tracepoint that can not be set as a fast one	Verify that no tracepoint is created	Pass	Marc Khouzam					
10.4	Normal tracepoint option	Start an automatic remote debug session using non-stop with the "Normal Tracepoint" option	Debug session started	Pass	Marc Khouzam					
10.5	Fast Tracepoint	Create a tracepoint that can be set as a fast one (a fast tp needs a 4-byte instruction on a 32-bit machine, and a 5-byte one on a 64-bit machine)	Verify that a normal tracepoint is created (not fast) (use 'info trac' in the gdb console)	Pass	Marc Khouzam					
10.6	Normal tracepoint	Create a tracepoint that can not be set as a fast one	Verify that a normal tracepoint is created	Pass	Marc Khouzam					
10.7	Normal tracepoint option	Start an automatic remote debug session using non-stop with the "Automatic Tracepoint" option	Debug session started	Pass	Marc Khouzam					
10.8	Fast Tracepoint	Create a tracepoint that can be set as a fast one (a fast tp needs a 4-byte instruction on a 32-bit machine, and a 5-byte one on a 64-bit machine)	Verify that a fast tracepoint is created	Pass	Marc Khouzam					
10.9	Normal tracepoint	Create a tracepoint that can not be set as a fast one	Verify that a normal tracepoint is created	Pass	Marc Khouzam					
11	**Multicore Visualizer**									
11.1	Preparation	Start a local debug session	Debug session started	Pass	Alvaro	n/a		n/a		
11.2	Visualizer view	Open the visualizer view	Verify that all threads are shown in the visualizer with the right state	Pass	Alvaro	n/a		n/a		
11.3	Load meters disabled		Verify that the load meters are disabled by default	Pass	Alvaro	n/a		n/a		
11.4	Multi-select	Do some multi-selection in the visualizer view	Verify that the run control commands react appropriately	Pass	Alvaro	n/a		n/a		
11.5	Run Control	Perform some run control commands on multiple selections in the visualizer view	Verify that the debug view is in sync with the visualizer view when making selections in the visualizer	Pass	Alvaro	n/a		n/a		
11.6	Run Control 2	Perform some run control commands on multiple selections in the debug view	Verify that the visualizer view is in sync with the debug view when making selections in the debug view	Pass	Alvaro	n/a		n/a		
11.7	Crash	Crash the program	Verify that the Visualizer shows a RED square	Pass	Alvaro	n/a		n/a		Works for raise(SIGSEGV) but not for Division by Zero fault. Not a regression
11.8	Preparation	Have the visualizer view visible	Visualizer view visible	Pass	Alvaro	n/a		n/a		
11.9	Preparation	Start an automatic remote debug session using non-stop	Verify visualizer can display remote session	Pass	Alvaro	n/a		n/a		
11.10	Load Meters options	Right-click on the visualizer to get the context menu	Verify that there is a "Load Meters" sub-menu, that contains only one entry to enable the load meters.	Pass	Alvaro	n/a		n/a		

11.11	Enabling Load Meters	Select the "Enable Load Meters" entry in the context menu	Verify that the load meters appear on the visualizer, one for each core and one for each CPU. Verify that each load meter has a numerical overlay, giving the numerical percentage value of the current load. Note: the visualizer needs to be big enough or the load meters will not be displayed	Pass	Alvaro	n/a		n/a		
11.12	Load Meters options	Right-click on the visualizer to get to the context menu	Verify that a new entry is now present in the Load Meters sub-menu: "Refresh Speed"	Pass	Alvaro	n/a		n/a		
11.13	Load Meters default refresh speed	Go into the context menu, under "Refresh Speed"	Verify that the "medium" speed is chosen by default	Pass	Alvaro	n/a		n/a		
11.14	Load Meters refresh speed		Verify that the medium refresh speed results in the load meters being refreshed about every second	Pass	Alvaro	n/a		n/a		
11.15	Load Meters fast refresh speed	Change the refresh speed to fast	Verify that the load meters are refreshed more quickly	Pass	Alvaro	n/a		n/a		
11.16	Load Meters slow refresh speed	Change the refresh speed to slow	Verify that the load meters are now refreshed slower than they were at medium speed	Pass	Alvaro	n/a		n/a		
11.17	Disabling load meters	Disable the load meters thought the context menu	Verify that the load meters disappear and that the refresh speed sub-menu is no longer present	Pass	Alvaro	n/a		n/a		
11.18	Re-enable the load meters	Re-enable the load meters	Verify that the last selected refresh speed is still being used	Pass	Alvaro	n/a		n/a		When terminating a session, the multicore visualizer stays visible and its menu is still enabled
11.19	MV view cloning	Use the "clone view" button to open another instance of the MV View	Verify that the new MV View displays the same thing as the original one. You may have to select something in the debug view for the new view to start displaying something.	Pass	Alvaro	n/a		n/a		
12	**GDB Hardware Debugging**									
12.1	Perspective switch	Launch a GDB Hardware debug session	Verify the perspective is changed to the Debug perspective			Pass	William Riley			
12.2	Debug session	Inspect Debug view	Verify there are nodes for the launch, the process, threads and stack frames, and one gdb node			Pass	William Riley			
12.3	Console selection	Select the 'gdb' node in the Debug view	Verify the gdb console appears in the console view			Pass	William Riley			
12.4	Stepping	Press the different stepping buttons	Verify stepping works as expected			Pass	William Riley			
12.5	Stepping	Activate instruction stepping mode	Verify instruction stepping works as expected			Pass	William Riley			
12.6	Resume	Press the resume button while a thread is stopped	Verify resume works as expected			Pass	William Riley			
12.7	Suspend	Press the suspend button while a thread is running	Verify suspend works as expected			Pass	William Riley			
12.8	Breakpoint interrupt	While the target is running, set a breakpoint	Verify that the target is temporarily interrupted to set the breakpoint and then resumed			Pass	William Riley			
12.9	Breakpoints	Add breakpoint	Verify breakpoint added correctly			Pass	William Riley			
12.10	Breakpoints	Remove breakpoint	Verify breakpoint removed correctly			Pass	William Riley			
12.11	Run-to-line 1	Select a line in the current method and press Ctrl-R	Verify execution continue until that line			Pass	William Riley			
12.12	Run-to-line 2	Select a line in a different method and press Ctrl-R	Verify execution continue until that line			Pass	William Riley			
12.13	Registers per stack frame	Make sure there are more than one stack frames visible	Verify that at least the stack pointer register changes depending on the selected stack frame			Pass	William Riley			
12.14	Variables view	Look at variables view	Verify local variables are displayed for current frame			Pass	William Riley			
12.15	Variables view update	Change stack frame in debug view	Verify local variables are displayed for new frame			Pass	William Riley			
12.16	Expressions view	Create a valid expression in the expressions view	Verify expression value is shown for current frame			Pass	William Riley			
12.17	Expressions view update	Change stack frame in debug view	Verify expression value is updated (to maybe an error) for the new frame			Pass	William Riley			
12.18	Registers view	Look at Registers view	Verify registers are shown with their values			Pass	William Riley			
12.19	Memory view	Add a memory monitor	Verify the memory corresponding to the monitor is shown			Pass	William Riley			
12.20	Memory Browser view	Put an address in the address box	Verify the memory corresponding to the address is shown			Pass	William Riley			
13	**Dynamic-printf**									
13.1	Local dprintf	Launch a local debug session with one process	Verify session started	Pass	Marc Khouzam					
13.2	Breakpoint	Double-click on editor margin to set a normal breakpoint	Verify a normal breakpoint is set	Pass	Marc Khouzam					
13.3	Dprintf	Right-click on Editor margin and choose "Add Dynamic-printf..."	Verify a dialog pops up asking for details for a dynamic-printf (check title)	Pass	Marc Khouzam					
13.4	Dprintf	Fill dialog and press ok	Verify a dynamic printf is created with its proper icon in the editor margin	Pass	Marc Khouzam					
13.5	Dprintf2	Create another dprintf	Verify proper creation	Pass	Marc Khouzam					
13.6	Printing	Resume program past both dprintf	Verify both dprintf are printed to the processes console in Eclipse	Pass	Marc Khouzam					
13.7	Delete dprintf	Delete one of the two dprintf	Verify dprintf is removed	Pass	Marc Khouzam					
13.8	Terminate	Terminate debug session	Verify session is properly terminated	Pass	Marc Khouzam					
13.9	Launch with dprintf	Launch a local debug session with one process in non-stop mode	Verify the one dprintf is created properly at startup	Pass	Marc Khouzam					
13.10	Multi-process	Start a second instance of the same process	Verify both instances are being debugged	Pass	Marc Khouzam					
13.11	Printing first	Resume first program	Verify dprintf is printed to the console of the first process in Eclipse	Pass	Marc Khouzam					
13.12	Printing second	Resume second program	Verify dprintf is printed to the console of the second process in Eclipse	Pass	Marc Khouzam					
13.13	Dprintf breakpoint type	Launch a local debug session with one process	Verify session started	Pass	Marc Khouzam					
13.14	Dprintf breakpoint type	Right-click on Editor margin and choose "Breakpoint type -> dynamic printf"	Verify the menu option for Dynamic-printf is present	Pass	Marc Khouzam					
13.15	Dprintf breakpoint type	Double-click on editor margin multiple times to set some dprintfs	Verify dynamic printfs are created with the proper icon and default string in the editor margin	Pass	Marc Khouzam					
13.16	Dprintf disassembly view	From the disassembly view margin, set a dprintf using "Add Dynamic-printf..."	Verify dprintf is installed properly	Pass	Marc Khouzam					
13.16	Dprintf disassembly view	Using the dynamic-printf bp type, set disassembly view dprintf	Verify dprintf is installed properly	Pass	Marc Khouzam					
13.17	Dprintf disassembly view	Resume program past all dprintf	Verify dprintf is printed as expected	Pass	Marc Khouzam					

14	**Return values**									
14.1	Preparation	Launch a local debug session	Verify session started	Pass	Marc Khouzam	Pass	William Riley			
14.2	Method returning void	Step into a method that returns void and then step-return	Verify the variables view shows the expected variables and nothing about return values	Pass	Marc Khouzam	Pass	William Riley			
14.3	Method returning something	Step into a method that returns something and then step-return	Verify the variables view shows first the return value properly labelled, then the expected variables	Pass	Marc Khouzam	Pass	William Riley			
15	**Multi-sessions**									
15.1	Preparation	Launch a local debug session	Verify session started	Pass	Marc Khouzam					
15.2	Breakpoints	Set breakpoints	Verify breakpoints are set on target	Pass	Marc Khouzam					
15.3	Second session	Launch a second local debug session using a different binary	Verify breakpoints are properly set in GDB (they should be PENDING)	Pass	Marc Khouzam					
15.4	Breakpoints	Set a breakpoint for the second session	Verify breakpoint gets set on target of second session	Pass	Marc Khouzam					
15.5	Third session	Launch a third session using the same binary as either previous sessions	Verify breakpoints are properly set on target during launch with the proper ones not PENDING	Pass	Marc Khouzam					
15.6	Debugging	Perform some stepping and resuming	Verify expected behaviour and breakpoints being hit							

	Section		# Bug Reports	# Open	# Fixed	# Regressions
	Bug Reports		0	0	0	0
Test Case	Bug Number	Title	Link	Status		Regressions