Yocto 1.3 M1 Fullpass Test

Test Report

Project: yocto

Author: admin

Printed by TestLink on 12/06/2012

1 Test Suite : Yocto 1.3 M1 Fullpass Test

1.1 Test Suite : ADT Toolchain

Test Case TC-2593: ADT installer Installation	
Summary: To check proper installation of ADT.	
Steps: 1.Get the tarball of adt_installer. 2.Edit configuration file adt_installer.conf, the parameter YOCTOADT_TARGET_SYSROOT_LOC_<arch> describes the location of the target sysroot on the development host, the location we call it SYSROOT in step 4, other paramaters pls refer to section 2.1.1.2.(http://www.yoctoproject.org/docs/current/adt-manual/adt-manual.html#using-the-adt-installer.) 3.Run script adt_installer to install by ./adt installer. 4.After the installation, setup cross compile environment by the commander source /opt/poky/{test_version}/environment-set-up-{target arch}-XXX. 5. Lauch the target enviroment by qemu: runqemu nfs KERNEL SYSROOT, (KERNEL is downloaded by adt installer scripte, you can find it in download_image folder in the adt installer workfolder). For example, my configuration set like this YOCTOADT_TARGETS=""x86"" and YOCTOADT_TARGET_SYSROOT_LOC_x86=""$HOME/test-yocto/x86"" , the connmand i run is : runqemu nfs ~/adt-installer/download_image/bzImage-qemux86.bin ~/test-yocto/x86. 6.Launch a terminal on target system, run ""uname -a"" to check the target architectures. Target Arch | ---------- | | | | | qemux86 qemux86-64 qemuarm qemuppc qemumips |- x86 yes yes yes yes yes Host Arch-| |- x86-64 yes yes yes yes yes Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all above.	
Expected Results: 1.No exception in installation. 2.Step 5 can launch normally. 3.The architectures is right as you set in adt_installer.	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2594: Cross-Toolchain Tarball Installation
Summary:

Cross-Toolchain Tarball Installation

Steps:

1.Get the tarball and find the folder that matches your host development system (i.e. i586 for 32-bit machines or x86_64 for 64-bit machines).

2.Go into that folder and download the toolchain tarball whose name includes the appropriate target architecture. For example,you are going to use your cross-toolchain for an Intel-based 32-bit target, go into the x86_64 folder and download the following tarball: XXX-eglibc-x86_64-i586-toolchain-{version}.tar.bz2.
3. Make sure you are in the root directory with root privileges and then expand the tarball. Once the tarball is expanded, the cross-toolchain is installed.
4.Setup cross compile environment by the commander source /opt/poky/{test_version}/environment-set-up-{target arch}-XXX.
5. Run command: runqemu nfs KERNEL SYSROOT(KERNEL and SYSROOT could be got from autobuilder or local build, the most convenient you can use the case of ADT installer Installation's KERNEL and SYSROOT, but which should be consistent with the target arch set by toolchain downloaded in above steps)

	Target Arch				
	qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch- |- x86	yes	yes	yes	yes	yes
Host Arch- |- x86-64	yes	yes	yes	yes	yes

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all above.

Expected Results:

Launch qemu with the right target architecture normally.

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2595: Using BitBake to build the toolchain

Summary:

Using BitBake to build the toolchain

Steps:

1.Prepare an existing Yocto Project build tree.
2.Set MACHINE variable in the local.conf file as the target architecture.
3.Source the environment setup script oe-init-build-env located in the Yocto Project files.
4.Run bitbake meta-ide-support to complete the cross-toolchain installation.

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, and both host architectures(32bit and 64bit) should be covered, no need cover all target architectures so select any arch as you like.
Expected Results: The tarball for the cross-toolchain is generated without error and it may work well.

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2596: gcc from ADT installer can build c program
Summary: gcc from ADT Installer can build c program and run with qemu-${ARCH} command or in target image

Steps:
 	
1. Install ADT installer and setup cross compile environment.
2. compile following program test.c "${CC} test.c -o test -lm"
3. run "test" with qemu-${ARCH} or run it into corresponding target image and check the output
#########
#include <stdio.h>
#include <math.h>
double
convert(long long l)
{
return (double)l; // or double(l)
}
int
main(int argc, char * argv[])
{
long long l = 10;
double f;
f = convert(l);
printf("convert: %lld => %f\n", l, f);
f = 1234.67;
printf("floorf(%f) = %f\n", f, floorf(f));
return 0;
}
#########
 	
Test Range: Three distributions Ubuntu, Fedora and OpenSUSE, two host architectures x86 and

x86_64, five target architectures qenux86,qemux86-64,qemuarm,qemuppc and qemumips, No need full covered just cross covered to test.

Expected Results:

executable binary test can run without problem

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2597: g++ from ADT installer can build c program

Summary:
g++ from ADT installer can build c program and run with qemu-${ARCH} command or in target image

Steps:

1. Install ADT installer and setup cross compile environment.
2. compile following program test.c "${CXX} test.c -o test -lm"
3. run "test" with qemu-${ARCH} or run it in corresponding target image and check the output

```
#########
#include <stdio.h>
#include <math.h>

double
convert(long long l)
{
  return (double)l;    // or double(l)
}
int
main(int argc, char * argv[])
{
  long long l = 10;
  double f;
  f = convert(l);
  printf("convert: %lld => %f\n", l, f);

  f = 1234.67;
  printf("floorf(%f) = %f\n", f, floorf(f));
  return 0;
}
#########
```

Test Range: Three distributions Ubuntu, Fedora and OpenSUSE, two host architectures x86 and x86_64, five target architectures qenux86,qemux86-64,qemuarm,qemuppc and qemumips, No need full covered just cross covered to test.

Expected Results: executable binary test can run without problem	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2598: ADT installer could build cvs project	
Steps: 1. Install ADT installer and setup cross compile environment 2. Download cvs project, http://ftp.gnu.org/non-gnu/cvs/source/feature/1.12.13/cvs-1.12.13.tar.bz2 3. With the cross compile environment, run "./configure ${CONFIGURE_FLAGS}", "make", "make install DESTDIR=/opt/tmp" Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below. Target Arch | ---------- | | | | | qemux86 qemux86-64 qemuarm qemuppc qemumips |- x86 yes yes yes yes yes Host Arch-| |- x86-64 yes yes yes yes yes	
Expected Results: cvs project could be compiled successfully with ADT toolchain	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2599: ADT installer could build iptables project
Summary:

ADT installer could build iptables project

Steps:

1. Install ADT installer and setup cross compile environment.
2. Download the latest version iptables project, now is iptables-1.4.13.
3. With the cross compile environment, run "./configure ${CONFIGURE_FLAGS}", "make", "make install DESTDIR=/opt/tmp"

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below.

	Target Arch			
	qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch- x86	yes	yes	yes	yes	yes
Host Arch- x86-64	yes	yes	yes	yes	yes

Expected Results:

iptables could be compiled successfully

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2600: ADT installer could build sudoku-savant project

Summary:

ADT installer could build sudoku-savant project

Steps:

1. Install ADT installer and setup cross compile environment.
2. Download sudoku-savant project, http://downloads.sourceforge.net/project/sudoku-savant/sudoku-savant/sudoku-savant-1.3/sudoku-savant-1.3.tar.bz2
3. With the cross compile environment, run "./configure ${CONFIGURE_FLAGS}", "make"

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below.

	Target Arch			

	qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch-| |- x86	yes	yes	yes	yes	yes
|- x86-64	yes	yes	yes	yes	yes

Expected Results:

sudoku-savant could be compiled successfully

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2601: gcc from meta-toolchain can build c program

Summary:

gcc from meta-toolchain can build c program and run with qemu-${ARCH} command or in target image

Steps:

1. Install toolchain tarball and setup cross compile environment.
2. compile following program test.c "${CC} test.c -o test -lm"
3. run "test" with qemu-${ARCH} or run it into corresponding target image and check the output.

#########

```
#include <stdio.h>
#include <math.h>
double
convert(long long l)
{
  return (double)l;    // or double(l)
}
int
main(int argc, char * argv[])
{
  long long l = 10;
  double f;
  f = convert(l);
  printf("convert: %lld => %f\n", l, f);
  f = 1234.67;
  printf("floorf(%f) = %f\n", f, floorf(f));
  return 0;
}
```

#########

Test Range: Three distributions Ubuntu, Fedora and OpenSUSE, two host architectures x86 and x86_64, five target architectures qenux86,qemux86-64,qemuarm,qemuppc and qemumips, No need full covered just cross covered to test.

Expected Results:

executable binary test can run without problem

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2602: g++ from meta-toolchain can build c program

Summary:

g++ from meta-toolchain can build c program and run with qemu-${ARCH} command or in target image

Steps:

1. Install toolchain tarball and setup cross compile environment.
2. compile following program test.c "${CXX} test.c -o test -lm"
3. run "test" with qemu-${ARCH} or run it in corresponding target image and check the output.

#########

```
#include <stdio.h>
#include <math.h>

double
convert(long long l)
{
  return (double)l;    // or double(l)
}
int
main(int argc, char * argv[])
{
  long long l = 10;
  double f;
  f = convert(l);
  printf("convert: %lld => %f\n", l, f);

  f = 1234.67;
  printf("floorf(%f) = %f\n", f, floorf(f));
  return 0;
}
```

#########

Test Range: Three distributions Ubuntu, Fedora and OpenSUSE, two host architectures x86 and x86_64, five target architectures qenux86,qemux86-64,qemuarm,qemuppc and qemumips, No need full covered just cross covered to test.

Expected Results: executable binary test can run without problem	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2603: meta-toolchain could build cvs project	
Summary: meta-toolchain could build cvs project	
Steps: 1. Install toolchain tarball and setup cross compile environment 2. Download cvs project, http://ftp.gnu.org/non-gnu/cvs/source/feature/1.12.13/cvs-1.12.13.tar.bz2 3. With the cross compile environment, run "./configure ${CONFIGURE_FLAGS}", "make", "make install DESTDIR=/opt/tmp" Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below. Target Arch | ---------- | | | | | qemux86 qemux86-64 qemuarm qemuppc qemumips |- x86 yes yes yes yes yes Host Arch-| |- x86-64 yes yes yes yes yes	
Expected Results: cvs project could be compiled successfully	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2604: meta-toolchain could build iptables project	
Summary: meta-toolchain could build iptables project	
Steps: 1. Install toolchain tarball and setup cross compile environment 2. Download iptables project, http://netfilter.org/projects/iptables/files/iptables-1.4.13.tar.bz2 3. With the cross compile environment, run "./configure ${CONFIGURE_FLAGS}", "make", "make install DESTDIR=/opt/tmp" Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below. Target Arch | ---------- | | | | | qemux86 qemux86-64 qemuarm qemuppc qemumips |- x86 yes yes yes yes yes Host Arch-| |- x86-64 yes yes yes yes yes	
Expected Results: iptables could be compiled successfully	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2605: meta-toolchain could build sudoku-savant project
Summary: sudoku-savant could be compiled with meta-toolchain
Steps: 1. Install toolchain tarball and setup cross compile environment 2. Download sudoku-savant project, http://downloads.sourceforge.net/project/sudoku-savant/sudoku-savant/sudoku-savant-1.3/sudoku-savant-1.3.tar.bz2 3. With the cross compile environment, run "./configure ${CONFIGURE_FLAGS}", "mak", "make install DESTDIR=/opt/tmp" Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below. Target Arch |

	qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch- |					
|- x86	yes	yes	yes	yes	yes
|- x86-64	yes	yes	yes	yes	yes

Expected Results:

sudoku-savant could be compiled successfully

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2606: Launch qemu by meta-toolchain

Summary:

Check if unfs works for qemu target by meta-toolchain

Steps:

1.Prepare a *rootfs.tar.bz2 image

2. Prepare a folder under poky directory as <rootfs-dir>, for example poky/temp
3. Install toolchain tarball and setup cross compile environment.
4. Run command "runqemu-extract-sdk *rootfs.tar.bz2 poky/temp"
5. Run command "runqemu nfs <kernel> <rootfs-dir>"

Test Range:Target architectures independent, so select any target arch with three distrobutions(Ubuntu, Fedora and OpenSUSE) and cover two host architetures(x86 x86_64).

Expected Results:

QEMU target should be started with unfs

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2607: Launch qemu by eclipse

Summary:

Eclipse can launch the target enviroment by adt-installer toolchain.

Steps:

1.Set the Yocto ADT's toolchain root location, sysroot location and kernel, in the menu Windows -> Preferences -> Yocto ADT.
(a)Point to the Toolchain: If you are using a stand-alone pre-built toolchain, you should be pointing to the /opt/poky/{test-version} directory as Toolchain Root Location, This is the location for toolchains installed by the ADT Installer or by hand.If you are using a system-derived toolchain, the path you provide for the Toolchain Root Location field is the Yocto Project's build directory.
(b)Specify the Sysroot Location: Sysroot Location is the location where the root filesystem for the target hardware is created on the development system by the ADT Installer(SYSROOT in step 2 of the case ADT installer Installation).
(c)Select the Target Architecture: The target architecture is the type of hardware you are going to use or emulate. Use the pull-down Target Architecture menu to make your selection.
(d) QEMU:Select this option if you will be using the QEMU emulator(KERNEL in step 5 of the case ADT installer Installation)
(e) select OK to save the settings.
2.In the eclpse toolbar, expose the Run -> External Tools menu. Your image should appear as a selectable menu item.
3.Select your image in the navigation pane to launch the emulator in a new window.
4.If needed, enter your host root password in the shell window at the prompt. This sets up a Tap 0 connection needed for running in user-space NFS mode.

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below.

		Target Arch			
		qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch-	x86	yes	yes	yes	yes	yes
	x86-64	yes	yes	yes	yes	yes

Expected Results:

Qemu can be lauched normally.

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2608: Launch qemu by Yocto build tree

Summary:

Check if unfs works for qemu target by toolchain from yocto build tree.

Steps:

1.Follow the steps of case "Using BitBake to build the toolchain".

2.Prepare a *rootfs.tar.bz2 image
3.Prepare a folder under poky directory as <rootfs-dir>.
4.Run command "runqemu-extract-sdk *rootfs.tar.bz2 <rootfs-dir>"
5. Run command "runqemu nfs <kernel> <rootfs-dir>"

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below.

		Target Arch			
		qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch	x86	yes	yes	yes	yes	yes
	x86-64	yes	yes	yes	yes	yes

Expected Results:

Qemu can be lauched normally.

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2609: C empty template

Summary:

C empty template works well with eclipse.

Steps:
1.Launch a qemu of target enviroment.(Reference to case Launch qemu by eclipse)
2.Select File -> New -> Project.
3.Double click C/C++.
4.Click C or C++ Project to create the project.
5.Expand Yocto ADT Project.
6.Select Empty Project.
7.Put a name in the Project name: field. Do not use hyphens as part of the name.
8.Click Next.
9.Add information in the Author and Copyright notice fields.
10.Click Finish.
11.If the "open perspective" prompt appears, click "Yes" so that you in the C/C++ perspective.
12.Add or import an existing project's code to the empty project.
13.Right click the project -> Build project.
14.Right click it again and Run as -> Run Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-run, then select Run button on the bottom right corner.
15.Right click it again and debug as -> Debug Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-debug, then select Debug button on the bottom right corner.

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below.

		Target Arch			
		qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch-	x86	yes	yes	yes	yes	yes
	x86-64	yes	yes	yes	yes	yes

Expected Results: Build succeed	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2610: Build ANSI C template
Summary: Eclipse can build and run C project which based on "Hello World ANSI C Autotools Project" template.
Steps: 1.Launch a qemu of target enviroment.(Reference to case Launch qemu by eclipse) 2.Select File -> New -> Project. 3.Double click C/C++. 4.Click C or C++ Project to create the project. 5.Expand Yocto ADT Project. 6.Select Hello World ANSI C Autotools Project. 7.Put a name in the Project name. Do not use hyphens as part of the name. 8.Click Next. 9.Add information in the Author and Copyright notice fields. 10.Click Finish. 11.If the "open perspective" prompt appears, click "Ye"" so that you in the C/C++ perspective. 12.Right click the project -> Build project. 13.Right click it again and Run as -> Run Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-run, then select Run button on the bottom right corner. 14.Right click it again and debug as -> Debug Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-debug, then select Debug button on the bottom right corner. Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below.

		Target Arch				
		qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch-	x86	yes	yes	yes	yes	yes
	x86-64	yes	yes	yes	yes	yes

Expected Results:

Build succeed and the console outputs "Hello world", you can also check the output on target.

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2611: Build Clutter C template

Summary:

Eclipse can build and run C project which based on "Clutter Hello world project" template.

Steps:
1.Launch a qemu of target enviroment.(Reference to case Launch qemu by eclipse)
2.Select File -> New -> Project.
3.Double click C/C++.
4.Click C Project to create the project.
5.Expand Yocto ADT Project.
6.Select Clutter Hello world project.
7.Put a name in the Project name: field. Do not use hyphens as part of the name.
8.Click Next.
9.Add information in the Author and Copyright notice fields.
10.Click Finish.
11.If the "open perspective" prompt appears, click "Yes" so that you in the C/C++ perspective.
12.Right click the project -> Build project.
13.Right click it again and Run as -> Run Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-run, then select Run button on the bottom right corner.
14.Right click it again and debug as -> Debug Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-debug, then select Debug button on the bottom right corner.

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below.

	Target Arch				
	qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch- | x86	yes	yes	yes	yes	yes
Host Arch- | x86-64	yes	yes	yes	yes	yes

Expected Results:

Build succeed and the console outputs "Hello world", we also check the output on target.

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips

image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2612: Build GTK C template
Summary: Eclipse can build and run C project which based on "Hello world GTK C Autotools Project" template.
Steps: 1.Launch a qemu of target enviroment.(Reference to case Launch qemu by eclipse) 2.Select File -> New -> Project. 3.Double click C/C++. 4.Click C Project to create the project. 5.Expand Yocto ADT Project. 6.Select Hello World GTK C Autotools Project. 7.Put a name in the Project name: field. Do not use hyphens as part of the name. 8.Click Next. 9.Add information in the Author and Copyright notice fields. 10.Click Finish. 11.If the "open perspective" prompt appears, click "Yes" so that you in the C/C++ perspective. 12.Right click the project -> Build project. 13.Right click it again and Run as -> Run Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-run, then select Run button on the bottom right corner. 14.Right click it again and debug as -> Debug Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-debug, then select Debug button on the bottom right corner. Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below. Target Arch | --- | | | | | qemux86 qemux86-64 qemuarm qemuppc qemumips |- x86 yes yes yes yes yes Host Arch-| |- x86-64 yes yes yes yes yes
Expected Results: Build succeed and the console outputs "Hello world", we also check the output on target.

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2613: C++ empty template
Summary:

C++ project which based on "Empty Project" template works well.

Steps:
1.Launch a qemu of target enviroment.(Reference to case Launch qemu by eclipse)
2.Select File -> New -> Project.
3.Double click C/C++.
4.Click C++ Project to create the project.
5.Expand Yocto ADT Project.
6.Select Empty Project.
7.Put a name in the Project name. Do not use hyphens as part of the name.
8.Click Next.
9.Add information in the Author and Copyright notice fields.
10.Click Finish.
11.If the "open perspective" prompt appears, click "Yes" so that you in the C/C++ perspective.
12.Add or import an existing project's code to the empty project.
13.Right click the project -> Build project.
14.Right click it again and Run as -> Run Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-run, then select Run button on the bottom right corner.
15.Right click it again and debug as -> Debug Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-debug, then select Debug button on the bottom right corner.

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below.

	Target Arch				
	qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch- | x86	yes	yes	yes	yes	yes
Host Arch- | x86-64	yes	yes	yes	yes	yes

Expected Results:

Build succeed

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2614: Build C++ autotool template

Summary:

Eclipse can build and run C++ project which based on "Hello world C++ Autotools Projec" template.

Steps:
1.Launch a qemu of target enviroment.(Reference to case Launch qemu by eclipse)
2.Select File -> New -> Project.
3.Double click C/C++.
4.Click C or C++ Project to create the project.
5.Expand Yocto ADT Project.
6.Select Hello World ANSI C Autotools Project/Empty Project/Clutter Hello world project/Hello

World GTK C Autotools Project. They are Autotools-based projects based on a Yocto Project template.
7.Put a name in the Project name: field. Do not use hyphens as part of the name.
8.Click Next.
9.Add information in the Author and Copyright notice fields.
10.Click Finish.
11.If the "open perspective" prompt appears, click "Yes" so that you in the C/C++ perspective.
12.Right click the project -> Build project.
13.Right click it again and Run as -> Run Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-run, then select Run button on the bottom right corner.
14.Right click it again and debug as -> Debug Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-debug, then select Debug button on the bottom right corner.

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below.

	Target Arch				
	qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch-|- x86	yes	yes	yes	yes	yes
Host Arch-|- x86-64	yes	yes	yes	yes	yes

Expected Results:

Build succeed and the console outputs "Hello world", you can also check the output on target.

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2615: Build C++ Clutter template

Summary:

Eclipse can build and run C++ project which based on "Clutter Hello world project" template.

Steps:
1.Launch a qemu of target enviroment.(Reference to case Launch qemu by eclipse)
2.Select File -> New -> Project.
3.Double click C/C++.
4.Click C or C++ Project to create the project.
5.Expand Yocto ADT Project.
6.Select Clutter Hello world Project.
7.Put a name in the Project name: field. Do not use hyphens as part of the name.
8.Click Next.
9.Add information in the Author and Copyright notice fields.
10.Click Finish.
11.If the "open perspective" prompt appears, click "Yes" so that you in the C/C++ perspective.
12.Right click the project -> Build project.
13.Right click it again and Run as -> Run Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-run, then select Run button on the bottom right corner.
14.Right click it again and debug as -> Debug Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for

example /home/root/test-debug, then select Debug button on the bottom right corner.

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below.

	Target Arch				
	qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch- x86	yes	yes	yes	yes	yes
Host Arch- x86-64	yes	yes	yes	yes	yes

Expected Results:

Build succeed and the console outputs "Hello world", we also check the output on target.

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2616: Change Yocot Project Settings

Summary:

Changing Yocot Project Settings works well.

Steps:

1.Select an existing project based on yocto projct ADT templates.

2.Select Project -> Change Yocto Project Settings: This selection brings up the Project Yocto Settings Dialog and allows you to make changes specific to an individual project.
3.Make your configurations for the project and click "OK".
4.Select Project -> Reconfigure Project.

Test Range: Three distrobutions(Ubuntu, Fedora and OpenSUSE) and two host architetures(x86 x86_64) should be tested.

Expected Results:

No error with reconfigure

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2617: C empty template from cross installation

Summary:

C empty template cooperates with meta-toochain and ADT installer's sysroot.

Steps:
1.Follow the case "Change Yocot Project Settings" , using tarball installation's toolchain and adt-installer's sysroot to launch qemu.
2.Follow the case "C empty template" to verify c empty template work well.

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below.

	Target Arch				
	qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch	x86	yes	yes	yes	yes	yes
	x86-64	yes	yes	yes	yes	yes

Expected Results:

Build succeed

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2618: ANSI C template from cross installation

Summary:

Eclipse can build and run C project which based on "Hello World ANSI C Autotools Project" template with meta- toochain and ADT installer's sysroot.

Steps:
1.Follow the case "Change Yocot Project Settings" , using tarball installation's toolchain and adt-installer's sysroot to launch qemu.
2.Follow the case "Build ANSI C template" to verify the template work well.

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below.

	Target Arch				
	qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch	x86	yes	yes	yes	yes	yes
	x86-64	yes	yes	yes	yes	yes

Expected Results:

Build succeed and the console outputs "Hello world", you can also check the output on target.

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2619: Clutter C template from cross installation

Summary:

Eclipse can build and run C project which based on "Clutter Hello world project" template with meta-toolchain and ADT installer's sysroot.

Steps:
1.Follow the case "Change Yocot Project Settings" , using tarball installation's toolchain and adt-installer's sysroot to launch qemu.
2.Follow the case "Build Clutter C template" to verify the template work well.

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below.

	Target Arch				
	qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch- x86	yes	yes	yes	yes	yes
Host Arch- x86-64	yes	yes	yes	yes	yes

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2620: GTK C template from cross installation

Summary:

Eclipse can build and run C project which based on "Hello world GTK C Autotools Project" template with meta-toolchain and ADT installer's sysroot.

Steps:
1.Follow the case "Change Yocot Project Settings" , using tarball installation's toolchain and adt-installer's sysroot to launch qemu.
2.Follow the case "Build GTK C template" to verify the template work well.

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below. Target Arch | ---------- | | | | | qemux86 qemux86-64 qemuarm qemuppc qemumips |- x86 yes yes yes yes yes Host Arch-| |- x86-64 yes yes yes yes yes
Expected Results: Build succeed and the console outputs "Hello world", we also check the output on target.

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2621: C++ empty template from cross installation
Summary: C++ empty template works well with eclipse by meta-toolchain and ADT installer's sysroot.
Steps: 1.Follow the case "Change Yocot Project Settings" , using tarball installation's toolchain and adt-installer's sysroot to launch qemu. 2.Follow the case "C++ empty template" to verify the template work well. Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below. Target Arch | ---------- | | | | | qemux86 qemux86-64 qemuarm qemuppc qemumips |- x86 yes yes yes yes yes Host Arch-| |- x86-64 yes yes yes yes yes
Expected Results: Build succeed

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk

Last Result	**Not Run**
Keywords:	None

Test Case TC-2622: C++ autotool template from cross installation	
Summary: Eclipse can build and run C++ project which based on "Hello world C++ Autotools Project" template with meta-toolchain and ADT installer's sysroot.	
Steps: 1.Follow the case "Change Yocot Project Settings" , using tarball installation's toolchain and adt-installer's sysroot to launch qemu. 2.Follow the case "Build C++ autotool template" to verify the template work well. Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below. Target Arch | --- | | | | | qemux86 qemux86-64 qemuarm qemuppc qemumips |- x86 yes yes yes yes yes Host Arch-| |- x86-64 yes yes yes yes yes	
Expected Results: Build succeed and the console outputs "Hello world", you can also check the output on target.	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2623: Clutter C++ template from cross installation	
Summary: Eclipse can build and run C++ project which based on "Clutter Hello world project" template with meta-oochain and ADT installer's sysroot.	
Steps: 1.Follow the case "Change Yocot Project Settings" , using tarball installation's toolchain and adt-installer's sysroot to launch qemu. 2.Follow the case "Build C++ Clutter template" to verify the template work well. Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below. Target Arch	

		Target Arch			
	qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch	x86	yes	yes	yes	yes	yes
	x86-64	yes	yes	yes	yes	yes

Expected Results:

Build succeed and the console outputs "Hello world", we also check the output on target.

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2624: C empty template by build tree installation

Summary:

C empty template works well with eclipse from build tree.

Steps:
1.Follow case "Using BitBake to build the toolchain" to generate toolchain.
2. Set Yocto Project's build directory as Toolchain Root Location in eclipse toolbar Window -> Preferences ->Yocto Project ADT,select Build system derived toolchian, Sysroot Location and QEMU Kernel set the adt installer's.
3.Launch a qemu of target enviroment.(Reference to case Launch qemu by eclipse)
4.Select File -> New -> Project.
5.Double click C/C++.
6.Click C Project to create the project.
7.Expand Yocto ADT Project.
8.Select Empty Project.
9.Put a name in the Project name. Do not use hyphens as part of the name.
10.Click Next.
11.Add information in the Author and Copyright notice fields.
12.Click Finish.
13.If the "open perspective" prompt appears, click "Yes" so that you in the C/C++ perspective.
14.Add or import an existing project's code to the empty project.
15.Right click the project -> Build project.
16.Right click it again and Run as -> Run Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-run, then select Run button on the bottom right corner.
17.Right click it again and debug as -> Debug Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-debug, then select Debug button on the bottom right corner.

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below.

		Target Arch			
	qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch	x86	yes	yes	yes	yes	yes
	x86-64	yes	yes	yes	yes	yes

Expected Results: C empty template works well	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2625: ANSI C template from build tree instllation	
Summary: Eclipse can build and run C project which based on "Hello World ANSI C Autotools Project" template.	
Steps: 1.Follow case "Using BitBake to build the toolchain" to generate toolchain. 2. Set Yocto Project's build directory as Toolchain Root Location in eclipse toolbar Window -> Preferences ->Yocto Project ADT,select Build system derived toolchian, Sysroot Location and QEMU Kernel set the adt installer's. 3.Launch a qemu of target enviroment.(Reference to case Launch qemu by eclipse) 4.Select File -> New -> Project. 5.Double click C/C++. 6.Click C Project to create the project. 7.Expand Yocto ADT Project. 8.Select Hello World ANSI C Autotools Project. 9.Put a name in the Project name. Do not use hyphens as part of the name. 10.Click Next. 11.Add information in the Author and Copyright notice fields. 12.Click Finish. 13.If the "open perspective" prompt appears, click "Yes" so that you in the C/C++ perspective. 14.Right click the project -> Build project. 15.Right click it again and Run as -> Run Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-run, then select Run button on the bottom right corner. 16.Right click it again and debug as -> Debug Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-debug, then select Debug button on the bottom right corner. Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below. Target Arch | ---------- | | | | | qemux86 qemux86-64 qemuarm qemuppc qemumips |- x86 yes yes yes yes yes Host Arch-| |- x86-64 yes yes yes yes yes	
Expected Results: Build succeed and the console outputs "Hello world", we also check the output on target.	
Test Execution Cycle Type:	Fullpass
Case Automation	Manual

Type:	
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2626: Clutter C template from build tree instllation

Summary:

Eclipse can build and run C project which based on "Clutter Hello world project" template.

Steps:

1.Follow case "Using BitBake to build the toolchain" to generate toolchain.

2. Set Yocto Project's build directory as Toolchain Root Location in eclipse toolbar Window -> Preferences ->Yocto Project ADT,select Build system derived toolchian, Sysroot Location and QEMU Kernel set the adt installer's.
3.Launch a qemu of target enviroment.(Reference to case Launch qemu by eclipse)
4.Select File -> New -> Project.
5.Double click C/C++.
6.Click C Project to create the project.
7.Expand Yocto ADT Project.
8.Select Clutter Hello world project.
9.Put a name in the Project name. Do not use hyphens as part of the name.
10.Click Next.
11.Add information in the Author and Copyright notice fields.
12.Click Finish.
13.If the "open perspective" prompt appears, click "Yes" so that you in the C/C++ perspective.
14.Right click the project -> Build project.
15.Right click it again and Run as -> Run Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-run, then select Run button on the bottom right corner.
16.Right click it again and debug as -> Debug Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-debug, then select Debug button on the bottom right corner.

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below.

		Target Arch				
		qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch	x86	yes	yes	yes	yes	yes
	x86-64	yes	yes	yes	yes	yes

Expected Results:

Build succeed and the console outputs "Hello world", we also check the output on target.

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk

Last Result	**Not Run**
Keywords:	None

Test Case TC-2627: GTK C template from build tree instllation	
Summary: Eclipse can build and run C project which based on "Hello world GTK C Autotools Project" template.	
Steps: 1.Follow case "Using BitBake to build the toolchain" to generate toolchain. 2. Set Yocto Project's build directory as Toolchain Root Location in eclipse toolbar Window -> Preferences ->Yocto Project ADT,select Build system derived toolchian, Sysroot Location and QEMU Kernel set the adt installer's. 3.Launch a qemu of target enviroment.(Reference to case Launch qemu by eclipse) 4.Select File -> New -> Project. 5.Double click C/C++. 6.Click C Project to create the project. 7.Expand Yocto ADT Project. 8.Select Hello world GTK C Autotools Project. 9.Put a name in the Project name. Do not use hyphens as part of the name. 10.Click Next. 11.Add information in the Author and Copyright notice fields. 12.Click Finish. 13.If the "open perspective" prompt appears, click "Yes" so that you in the C/C++ perspective. 14.Right click the project -> Build project. 15.Right click it again and Run as -> Run Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-run, then select Run button on the bottom right corner. 16.Right click it again and debug as -> Debug Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-debug, then select Debug button on the bottom right corner. Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below. Target Arch | --- | | | | | qemux86 qemux86-64 qemuarm qemuppc qemumips |- x86 yes yes yes yes yes Host Arch-| |- x86-64 yes yes yes yes yes	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2628: C++ empty template from build tree installation
Summary:

C++ empty template works well with eclipse.

Steps:

1.Follow case "Using BitBake to build the toolchain" to generate toolchain.

2. Set Yocto Project's build directory as Toolchain Root Location in eclipse toolbar Window -> Preferences ->Yocto Project ADT,select Build system derived toolchian, Sysroot Location and QEMU Kernel set the adt installer's.
3.Launch a qemu of target enviroment.(Reference to case Launch qemu by eclipse)
4.Select File -> New -> Project.
5.Double click C/C++.
6.Click C++ Project to create the project.
7.Expand Yocto ADT Project.
8.Select Empty Project.
9.Put a name in the Project name. Do not use hyphens as part of the name.
10.Click Next.
11.Add information in the Author and Copyright notice fields.
12.Click Finish.
13.If the "open perspective" prompt appears, click "Yes" so that you in the C/C++ perspective.
14.Add or import an existing project's code to the empty project.
15.Right click the project -> Build project.
16.Right click it again and Run as -> Run Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-run, then select Run button on the bottom right corner.
17.Right click it again and debug as -> Debug Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-debug, then select Debug button on the bottom right corner.

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below.

	Target Arch				
	qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch- x86	yes	yes	yes	yes	yes
Host Arch- x86-64	yes	yes	yes	yes	yes

Expected Results:

C++ empty template works well

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2629: C++ autotool template from build tree instllation

Summary:

Eclipse can build and run C++ project which based on "Hello world C++ Autotools Project" template.

Steps:

1.Follow case "Using BitBake to build the toolchain" to generate toolchain.

2. Set Yocto Project's build directory as Toolchain Root Location in eclipse toolbar Window -> Preferences ->Yocto Project ADT,select Build system derived toolchian, Sysroot Location and QEMU Kernel set the adt installer's.
3.Launch a qemu of target enviroment.(Reference to case Launch qemu by eclipse)
4.Select File -> New -> Project.
5.Double click C/C++.
6.Click C++ Project to create the project.
7.Expand Yocto ADT Project.
8.Select Hello world C++ Autotools Project
9.Put a name in the Project name. Do not use hyphens as part of the name.
10.Click Next.
11.Add information in the Author and Copyright notice fields.
12.Click Finish.
13.If the "open perspective" prompt appears, click "Yes" so that you in the C/C++ perspective.
14.Right click the project -> Build project.
15.Right click it again and Run as -> Run Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-run, then select Run button on the bottom right corner.
16.Right click it again and debug as -> Debug Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-debug, then select Debug button on the bottom right corner.

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below.

	Target Arch				
	qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch- x86	yes	yes	yes	yes	yes
Host Arch- x86-64	yes	yes	yes	yes	yes

Expected Results:

Build succeed and the console outputs "Hello world", we also check the output on target.

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2630: Clutter C++ template from build tree instllation

Summary:

Eclipse can build and run C++ project which based on "Clutter Hello world project" template.

Steps:

1.Follow case "Using BitBake to build the toolchain" to generate toolchain.

2. Set Yocto Project's build directory as Toolchain Root Location in eclipse toolbar Window -> Preferences ->Yocto Project ADT,select Build system derived toolchian, Sysroot Location and

QEMU Kernel set the adt installer's.
3.Launch a qemu of target enviroment.(Reference to case Launch qemu by eclipse)
4.Select File -> New -> Project.
5.Double click C/C++.
6.Click C++ Project to create the project.
7.Expand Yocto ADT Project.
8.SelectClutter Hello world project.
9.Put a name in the Project name. Do not use hyphens as part of the name.
10.Click Next.
11.Add information in the Author and Copyright notice fields.
12.Click Finish.
13.If the "open perspective" prompt appears, click "Yes" so that you in the C/C++ perspective.
14.Right click the project -> Build project.
15.Right click it again and Run as -> Run Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-run, then select Run button on the bottom right corner.
16.Right click it again and debug as -> Debug Configurations..., then double click C/C++ Remote Application to new a configuration ,input Remote Absolute File path for C/C++ Application, for example /home/root/test-debug, then select Debug button on the bottom right corner.

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered, It has relationship both with host and target architectures , so test all below.

	Target Arch				
	qemux86	qemux86-64	qemuarm	qemuppc	qemumips
Host Arch- x86	yes	yes	yes	yes	yes
Host Arch- x86-64	yes	yes	yes	yes	yes

Expected Results:

Build succeed and the console outputs "Hello world", we also check the output on target.

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2631: Oprofile

Summary:

Oprofile can connect,start,stop and view the information.

Steps:

1.Install oprofile and oprofile-viewer on host machine,pls refer to wiki:https://wiki.yoctoproject.org/wiki/How_to_setup_environment_for_ADT_with_1.1_on_Fedora_16#Running_User-Space_Tools.
2.Launch a qemu of target enviroment.(Reference to case Launch qemu by eclipse)
3.Expose YoctoTools -> Oprofile to connect the target.
4.In Oprofile Viewer, it can connect, disconnect, start, stop, download and reset to the target.

Test Range:Three distributions Ubuntu, Fedora and OpenSUSE should be covered,Both x86 and x86-64

host architectures and five target architectures(qemux86, qemux86_64, qemuarm, qemuppc, qemumips) should be cross coverd.

	Target Arch				
Host Arch	qemux86	qemux86-64	qemuarm	qemuppc	qemumips
x86/x86-64	yes	yes	yes	yes	yes

Expected Results: Oprofile works well.	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2632: Perf

Summary:

Perf monitors the system's performance counter registers well.

Steps:

1.Launch a qemu of target enviroment.(Reference to case Launch qemu by eclipse)
2.Expose YoctoTools -> Perf to connect the target.
3.It will lauch a terminal for target, run "perf top" , and something should show up."

Test Range: Host arch independence, so select any one host from x86 and x86-64 with the target architectures qemux86_32, qemux86_64, qemuarm, qemuppc and qemumips.

	Target Arch				
Host Arch	qemux86	qemux86-64	qemuarm	qemuppc	qemumips
x86/x86-64	yes	yes	yes	yes	yes

Expected Results: Perf works well.	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual

Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2633: LTTng-user space	
Summary: Eclipse lttng-viewer should show the lttng trace information.	

Steps:
 	
1.Upgrade Eclipse Linux tools lttng to 0.4.0 which is under http://download.eclipse.org/technology/linuxtools/update.
 	
2.Install Lttng parser library 2.5 and 2.6 with the URL http://wiki.eclipse.org/Linux_Tools_Project/LTTng.
 	
3. In Eclipse, File->New Project and create a new Lttng project.
 	
4.Scp the file lttng.c to target machine, then in remote target run: gcc -lust lttng.c, it will generate a binary file named a.out.
5. Run lttng-ust YoctoProjectTools -> lttng-user space , in the window set the import project to your newly created lttng project, set Application as the absolute patch of a.out on target machine.
6.After lttng-ust is done, you should see an entry created in your ust project's Traces sub-dir, named as a long numbers.
7.Right click the entry to change the type to kernel type and select Open, then you may see some tracing date in the lttng-viewer.
########
#include <ust/marker.h>
 	
int main(int argc, char **argv)
{
int v=6;
char *st="lttng example";
 	
trace_mark(main, myevent, "firstarg %d secondarg %s", v, st);
 	
trace_mark(main, myotherevent, MARK_NOARGS);
 	
return 0;
 	
}
########	

Expected Results: LTTng viewer shows tracing data.	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk

Last Result	**Not Run**
Keywords:	None

Test Case TC-2634: PowerTop	
Summary: Eclipse can runs powertop on the remote target machine	
Steps: 1.Launch a qemu of target enviroment.(Reference to case Launch qemu by eclipse) 2.Expose YoctoTools -> PowerTop to connect the target. 3.It will new a tab named PowerTop to show the power usage information. Test Range: Host arch independence, so select any one host from x86 and x86-64 with the target architectures qemux86_32, qemux86_64, qemuarm, qemuppc and qemumips. Target Arch | ---------- | | | | | Host Arch qemux86 qemux86-64 qemuarm qemuppc qemumips | - x86/x86-64 yes yes yes yes yes	
Expected Results: PowerTop works well.	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2635: latencyTop	
Summary: Plug-in latencyTop can identify system latency.	
Steps: 1.Launch a qemu of target enviroment.(Reference to case Launch qemu by eclipse) 2.Expose YoctoTools -> PowerTop to connect the target. 3.It will new a terminal tab window to list what cause the system latency. 4.The information will be refesh every several seconds. Test Range: Host arch independence, so select any one host from x86 and x86-64 with the target architectures qemux86_32, qemux86_64, qemuarm, qemuppc and qemumips. Target Arch | ----------	

Host Arch | -	qemux86	qemux86-64	qemuarm	qemuppc	qemumips
x86/x86-64	yes	yes	yes	yes	yes

Expected Results:

latencyTop can identify system latency.

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2636: SystemTap

Summary:

SystemTap works well in eclipse.

Steps:

1.Refer to https://wiki.yoctoproject.org/wiki/Tracing_and_Profiling, work out a Kernel Module.

2.Expose YoctoProjectTools-> systemtap, Connect to the remote target, then Brower the kernel module from step 1.
3.Then the eclipse will new a terminal tab window and it shows :
export TERM=vt100;staprun /tmp/trace_open.ko
root@qemux86:/# export TERM=vt100;staprun /tmp/trace_open.ko
4.After that you can run it by the command staprun /tmp/trace_open.ko or crosstap trace_open.stp root@192.168.7.2 on host,then it will output :
ls(743) open ("/etc/ld.so.cache", O_RDONLY)
ls(743) open ("/lib/librt.so.1", O_RDONLY)
ls(743) open ("/lib/libcap.so.2", O_RDONLY)
ls(743) open ("/lib/libc.so.6", O_RDONLY)
ls(743) open ("/lib/libpthread.so.0", O_RDONLY)
ls(743) open (".",
O_RDONLY|O_CLOEXEC|O_DIRECTORY|O_LARGEFILE|O_NONBLOCK|O_CLOEXEC)

Test Range:Cause systemTap doesn't support qemumips, so we just cross test four target architectrues with two host arches.

Expected Results:

Target can run the module normally

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2637: Bitbake Commander	
Summary: Eclipse can create customized recipe by bitbake commander.	
Steps: 1.Swich to bitbake commander perspective on the right corner of eclipse window.. 2.New a project ->Yocto Bitbake Commander -> New Yocto Project , input the Project Name and Project Location, if you want to clone from Yocto Git Repository you may select the check box or import a existing yocto project build tree: File-> Import->Existing Projects into Workspace ->click Next ->Select root directory -> Brower… to select the project, after that the project will appear in Projects blank space,select it and click Finish. 3.Select the bitbake project, File -> New -> Yocto BitBake Commander -> BitBake Recipe, for remote archive packages, after you enter the src_url and click on "populate", it should calculate the archive md4, sha256, license checksum and auto generated recipe file name. 4.For local source packages, after entering file:///absolute path to the package, then populate, it should calc license checksum value and come up recipe name based on the package directory name.For example,you may add a recipe in poky-contrib tree, the name:m4-1.4.9, SRC_URL is ftp://ftp.gnu.org/gnu/m4/m4-1.4.9.tar.gz and add its Description, then "Populate" other infomations of the recipe. Test Range: It no need to connect to target, so it is target independent, just test with two host arches on three distrobutions(Ubuntu, Fedora and OpenSUSE).	
Expected Results: Recipes can be created.	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2638: Hob
Summary: Eclipse can launch hob.
Steps: 1. Clone a yocto project build tree. 2. File->New->Project->Yocot Project BitBake Commander->New Yocto project, click Next and give a name of the project, select the git folder as Project Location. 3.Select the project, in eclipse toolbar, Project -> Lauch HOB. 4.Select any Bitbake build directory which has compiled pseudo-native. Test Range: It no need to connect to target, so it is target independent, just test with two host

arches on three distrobutions(Ubuntu, Fedora and OpenSUSE).
Expected Results: Hob can be launched.

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2639: Hovors tooltip in bitbake commander
Summary: Hovors tooltip in bitbake commander
Steps: 1.New a bitbake recipe by populated the relevant information(include LICENSE field). 2.Modify the recipe using the variable reference, something like this:FOO = "foo" BAR = "${MACHINE}-${FOO} BAA="${BAR}", then, move the mouse over "${FOO}" and wait for a while to see if there is text hover information showing the value of variable "FOO". Move the mouse over ${MACHINE} , ${BAR}and see what's the value of that variable. 3.Modify the MACHINE variable in the file build/conf/local.conf under the BC project directory, save it. Move the mouse over ${MACHINE} and ${BAR} again to check if the text hover information has been changed to reflect your latest modification. 4.Modify the recipe add a bad line like"inherit aaa", the console should report some error infomations, and hovers tooltip doesn't work. Test Range: It no need to connect to target, so it is target independent, just test with two host arches on three distrobutions(Ubuntu, Fedora and OpenSUSE).
Expected Results: Hovers tooltip may show and change along with the configuration file.

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	undecided
target:	qemux86_32
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2640: Headless Build	
Summary: Headless build is attempt to imitate autobuild's action avoid to building failure before milestone testing. It is a way to install eclipse in commander line.	
Steps: 1.git clone eclipse-poky source and checkout the testing version,git clone git://git.yoctoproject.org/eclipse-poky . 2.Go to the source, modify script/setup.sh's proxy to make sure your network may approach the internet. 3.Run the script setup.sh, it will take a while to download and install the eclipse and other plug-in. 4.Step 3 will prompt a commander to tell you how to build it ,"ECLIPSE_HOME=/home/tester/git-work/eclipse-poky/eclipse scripts/build.sh <branch name> <release name>" to build, for example, we build milestone 1.2M4, then i run ECLIPSE_HOME=/home/tester/git-work/eclipse-poky/eclipse scripts/build.sh 1.2_M4 myEclipse1.2M4 5.After step 4, it will generate two tarball org.yocto.sdk-myeclipse1.2M4-201204051002-archive.zip and org.yocto.sdk-myeclipse1.2M4-201204051002.zip. 6.Install org.yocto.sdk-myeclipse1.2M4-201204051002-archive.zip with eclipse to make sure it can work well. We should test on 32 bit and 64 bit machine,including Opensese distro, before milestone testing.	
Expected Results: Eclipse may install successful and the plug-in built from it may work.	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Auto
Case State:	Ready
Feature:	sdk
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2641: Identify arch sysroot in meta-toolchain	
Summary: The eclipse may identify which architecture's sysroot through the target architecture selected.	
Steps: 1.Install toolchainSDK using untar tarballs. 2.Set /opt/poky/${VERSION}/sysroot as sysroot location, and toolchain root location, target architecture and kernel, in the menu Windows -> Preferences -> Yocto ADT. 3.New a C/C++ project based on Hello world autotools project. 4.Cross build and run the project.	

Expected Results: The project should build pass and print out Hello world in eclipse console.	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Auto
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2859: Yocto-BSP tool	
Summary: yocto-bsp tool can create BSP layer in eclipse	
Steps: 1.In eclipse, YoctoProjectTools->yocto-bsp, set Meta_data loaction as poky tree folder, such as /home/build/gitwork/poky, build location set any non-exist folder you want , set a BSP name as you like, for example myqemux86, BSP output location set any non-exist folder, for example i set /home/build/gitwok/poky/myqemux86, then select the arch and qemu arch. 2.Click Next, select kernel and kernel branch, then finish, it will create the BSP	
Expected Results: yocto-bsp tool can create an BSP layer.	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Auto
Case State:	Ready
Feature:	undecided
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
Last Result	**Not Run**
Keywords:	None

1.2 Test Suite : hob

Test Case TC-2642: hob launch without error
Summary: hob could be launched without error
Steps:

1. Prepare poky build environment 2. launch hob with command "hob" 3. Check if hob is launched correctly and no error message in console
Expected Results: hob launched correctly and no error message

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2643: add layer for new target build
Summary: user could add layer for new target build
Steps: 1. launch hob 2. click "icon" for "Layers", then choose one layer, for example, you could download meta-intel.git and add it into layers 3. check "Machine" list and sugarbay should be available
Expected Results: user could add layer for new target build

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2644: user could delete layer
Summary: user could delete layer
Steps: 1. launch hob 2. click "icon" for "Layers", then choose one layer, for example, you could download meta-intel.git and add it into layers 3. check "Machine" list and sugarbay should be available

4. click "Layers" again and delete meta-intel and meta-sugarbay from "Layers" 5. check "Machine" list and sugarbay should be removed	
Expected Results: user could delete layer	
Test Execution Cycle Type:	
Case Automation Type:	
Case State:	
Feature:	undecided
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2645: base image selection	
Summary: recipe list should be loaded for base image selection	
Steps: 1. launch hob 2. select one "Machine", for example, qemumips 3. choose one "Base image", for example, core-image-sato 4. click the icon for "View Recipes", there should be a list of recipes shown as selected	
Expected Results: recipe list should be loaded for base image selection	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2646: recipes parsing stop	
Summary: User could use "stop" button to stop recipes parsing	
Steps: 1. launch hob 2. select one "Machine", for example, qemuarm 3. when hob is parsing recipes, click "stop" button to abort the parse 4. choose another machine, for example, qemux86	
Expected Results:	

"stop" button could be used to abort recipes parsing

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2647: recipe list re-load for "base image" change
Summary: recipe list should be re-loaded if changing image type for "base image"
Steps: 1. launch hob 2. select one "Machine", for example, qemumips 3. choose one "Base image", for example, core-image-sato 4. click the icon for "View Recipes", there should be a list of recipes shown as selected 5. change the "Base image" to another type, for example, "core-image-minimal", the list of recipes should be re-loaded
Expected Results: recipe list should be re-loaded and totoal number of included recipes should be changed if changing image type for "base image"

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2648: recipe list re-load for "Machine" change
Summary: recipe list for should be re-loaded and correct when "Machine" changing
Steps: 1. launch hob 2. select one "Machine", for example, qemumips 3. choose one "Base image", for example, core-image-sato 4. click the icon for "View Recipes", there should be a list of recipes shown as selected 5. change the selection for "Machine", for example, qemux86 6. click the icon for "View Recipes", there should be a new list of recipes shown as selected
Expected Results:

recipe list should be re-loaded and included recipe number should be changed when "Machine" changing	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2649: No native recipe shown in recipe list	
Summary: There should be no native recipe shown in recipe list	
Steps: 1. launch hob 2. select one "Machine", for example, qemumips 3. click the icon for "View Recipes", check if there is any -native recipe shown	
Expected Results: There should be no native recipe shown in recipe list	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2650: search recipe name in recipe list	
Summary: User could search recipe name from "Search recipes"	
Steps: 1. launch hob 2. select one "Machine", for example, qemumips 3. click the icon for "View Recipes", then search recipe via "Search recipes" 4. the searched recipe should be shown up	
Expected Results: User could search recipe name from "Search recipes"	
Test Execution Cycle Type:	Weekly

Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2651: task list re-load when base image change	
Summary: task list for should be re-loaded when base image changing	
Steps: 1. launch hob 2. select one "Machine", for example, qemumips 3. choose one "Base image", for example, core-image-sato 4. click the icon for "View Recipes"->"Tasks", there should be a list of tasks shown as selected 5. change the selection for "Base image", for example, core-image-lsb 6. click the icon for "View Recipes", there should be a new list of tasks shown as selected	
Expected Results: task list for "recipe collections" and the number of selected tasks should be re-loaded when base image changing	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2652: brought in by dialog for recipes	
Summary: User could checked detailed list of brought in by information with dialog	
Steps: 1. launch hob 2. select one "Machine", for example, qemumips 3. choose one "Base image", for example, core-image-sato 4. click the icon for "View Recipes", there should be a list of recipes shown as selected 5. double click recipes, there should be a dialog popup, which shows all the recipes bring the slected recipe in	
Expected Results: the brought in by dialog could work well	
Test Execution	Weekly

Cycle Type:	
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2653: user could customize threads of bitbake and make	
Summary: user could customize threads of bitbake and make in hob	
Steps: 1. launch hob 2. select one "Machine", for example, qemux86 3. click "Advanced Settings", set "BB_NUMBER_THREADS" and "PARALLEL_MAKE" to 1, then click "Save" 4. select one image for "Base image", for example, "core-image-basic" 5. click "Build image" and check 'ps' command output if there is one thread running	
Expected Results: user could customize threads of bitbake and make in hob	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2654: progress bar to show build tasks left	
Summary: there should be a progress bar to show build tasks left	
Steps: 1. launch hob 2. select one "Machine", for example, qemux86 3. choose one "Base image", for example, core-image-minimal 4. click "Just bake" and there should be a progress bar to show the build tasks left	
Expected Results: there should be a progress bar to show build tasks left	
Test Execution Cycle Type:	Weekly

Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2655: ipk package build for image/package build	
Summary: build image with ipk package format	
Steps: 1. launch hob 2. select one "Machine", for example, qemux86 3. in "Settings"->"Output", select ipk for "packaging format" 4. click "Save" and select one image, for example, "core-image-basic" 5. click "Just bake" button and it should build recipes with ipk format	
Expected Results: build image with ipk package format	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2656: deb package build for image/package build	
Summary: build image with deb package format	
Steps: 1. launch hob 2. select one "Machine", for example, qemux86 3. in "Settings"->"Output", select deb for "packaging format" 4. click "Save" and select one image, for example, "core-image-basic" 5. click "Just Bake" button and it should build recipes with deb format	
Expected Results: build image with deb package format	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual

Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2657: rpm package build for image/package build
Summary: build image with rpm package format
Steps: 1. launch hob 2. select one "Machine", for example, qemux86 3. in "Settings"->"Output", select rpm for "packaging format" 4. click "Save" and select one image, for example, "core-image-basic" 5. click "Just bake" button and it should build recipes with rpm format
Expected Results: build image with rpm package format

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2658: multiple package format set for build
Summary: build image with multiple package format set
Steps: 1. launch hob 2. select one "Machine", for example, qemux86 3. in "Settings"->"Output", select all 3 options for "packaging format", rpm, ipk, deb 4. click "Save" and select one image, for example, "core-image-basic" 5. click "Just bake" button and it should build recipes with rpm, ipk, deb format
Expected Results: build image with multiple package format set

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob

target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2659: stop build during image/package building	
Summary: "stop build" button should be able to stop/force stop building	
Steps: 1. launch hob 2. select one "Machine", for example, qemuarm 3. choose one "Base image", for example, core-image-sato 4. click "Just bake" button and it should show a build progress bar 5. click "stop" button, then click "stop" or "force stop" to stop the build	
Expected Results: "stop build" button should be able to stop/force stop building	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2660: Tab view for "Building packges"	
Summary: Different tabs in "Building packages" should show Configuration, Issues and Log	
Steps: 1. launch hob 2. select one "Machine", for example, qemuarm 3. choose one "Base image", for example, core-image-sato 4. click "Just bake" button and it should go to build the image 5. check the tabs in "Building packages" page, there are 3 tabs - "Build Configuration" for configuration information, "Issues" for error/execption reported during build, "Log" for full log during build	
Expected Results: Different tabs in "Building packages" should show Configuration, Issues and Log	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob

target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2661: template file save/load	
Summary: user could save customized template file and load it in hob	
Steps: 1. launch hob 2. select one "Machine", for example, qemuarm 3. choose one "Base image", for example, core-image-basic 4. click "Build packages" and wait for it finished 5. click the icon for "View Packages", there should be a list of packages shown as selected, select some un-selected package, for example, acpid 6. de-select some selected package, for example, zypper 7. click "Build image" button and it should show a build progress bar 8. after build finished successfully, click the "Save Template Files" button to save the build information into a template file 9. re-launch hob and click "Templates" and choose the template file saved as above 10. The user customized recipe list should be shown in "View Recipes" and the added/removed information should be correct	
Expected Results: user could save customized template file and load it in hob	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2662: another build after stop build
Summary: user could start another build after stop a build
Steps: 1. launch hob 2. select one "Machine", for example, qemuarm 3. choose one "Base image", for example, core-image-sato 4. click "build image" button and it should show a build progress bar 5. click "stop" button, then click "stop" or "force stop" to stop the build 6. select another machine, for example, qemumips and choose another base image 7. click "build image" and wait for build finished
Expected Results:

user could start another build after stop a build

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2663: build a image without error(base image)
Summary: user could use hob to build a image without error
Steps: 1. launch hob 2. select one "Machine", for example, qemuarm 3. choose one "Base image", for example, core-image-minimal 4. click "Just bake" button and wait for a successful build finished
Expected Results: user could use hob to build a image without error

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2664: build a image without error (added recipe)
Summary: user could use hob to build a image without error
Steps: 1. launch hob 2. select one "Machine", for example, qemuarm 3. choose one "Base image", for example, core-image-minimal 4. click the icon for "View Recipes", there should be a list of recipes shown as selected, select some un-selected recipe, for example, acpid 5. click "build packages" and wait for a successful build 6. select acpid in "View packages" and click "Build image" 7. after build finished, check if the added recipe built into image
Expected Results:

user could use hob to build a image without error

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2665: build a image without error (remove recipe)
Summary: user could use hob to build a image without error
Steps: 1. launch hob 2. select one "Machine", for example, qemuarm 3. choose one "Base image", for example, core-image-sato 4. click "Build packages", and wait for a successful build 5. click the icon for "View Packages", there should be a list of packages shown as selected, deselect some selected package, for example, zypper 6. click "Build image" button and wait for a successful build finished 7. after build finished, check if the removed recipe not built into image
Expected Results: user could use hob to build a image without error

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2666: Run image in Hob
Summary: User could run image in Hob
Steps: 1. launch hob 2. select one "Machine", for example, qemuppc 3. choose one "Base image", for example, core-image-minimal 4. click "Just bake" button and it should go to build the image 5. after build is finished, slect the ext3 image and click "Run image", then choose one kernel for the

image, the qemu target will be launched	
Expected Results: User could start image in Hob via "Run image" botton	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2667: Deploy image in Hob	
Summary: User could deploy hddimg into USB stick	
Steps: 1. launch hob 2. add a layer for meta-intel BSP, for example, sugarbay, and choose sugarbay as Machine, choose "core-image-minimal" for "Base image" 3. in "Settings", make sure "live" is selected for "Image types" 5. click "Just bake" button and wait for a successful build finished 6. after build finished, choose "Deploy image" and insert a USB stick to burn the image into the stick 7. Use the stick to live boot on a real board	
Expected Results: User could deploy hddimg into USB stick	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2668: toolchain built correct with user customization
Summary: toolchain generated correct with user selection
Steps: 1. launch hob 2. select one "Machine", for example, qemumips

3. choose one "Base image", for example, core-image-sato 4. click icon for "Settings", and select "Build Toolchain", for toolchain host, you could pick up one, for example, x86_64 5. click "Just bake" button and wait for a successful build finished 6. after build finished, check if toolchain is built out with the correct host/target arch, and then use the toolchain to start the image built out by Hob	
Expected Results: toolchain generated correct with user selection	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2669: non-GPLv3 build	
Summary: non-GPLv3 build should be supported for hob	
Steps: 1. launch hob 2. select one "Machine", for example, qemumips 3. choose one "Base image", for example, core-image-minimal or core-image-basic 4. click icon for "Settings", and select "Exclue GPLv3 packages" 5. click "Just bake" button and wait for a successful build finished 6. check if there is any non-GPLv3 packages built in	
Expected Results: non-GPLv3 build should be supported for hob	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2670: distribution selection for build
Summary: user could select different distribution for "distribution"
Steps: 1. launch hob

2. select one "Machine", for example, qemumips 3. choose one "Base image", for example, core-image-minimal 4. click icon for "Settings"->"Build environment", and select different distribution for "Select Distro", for example, poky-lsb 5. click "build image" button and wait for a successful build finished	
Expected Results: user could select different distribution for "distribution"	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2671: package size shown	
Summary: detailed package size should be shown after build is finished	
Steps: 1. launch hob 2. select one "Machine", for example, qemumips 3. choose one "Base image", for example, core-image-minimal 4. click "Just bake" button and wait for a successful build finished 5. after that, the image size will be shown and you could check size of each package via "Edit packages"	
Expected Results: detailed package size should be shown after build is finished	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2672: recipe add/remove
Summary: user could add/remove recipes with correct information shown up in hob
Steps: 1. launch hob

2. select one "Machine", for example, qemumips 3. choose one "Base image", for example, core-image-minimal 4. click icon of "View Recipes" and it will show a list of recipes 5. select some un-selected recipes and de-select some selected recipes 6. check if the recipe number and dependency is correct	
Expected Results: user could add/remove recipes with correct information shown up in hob	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2673: package add/remove	
Summary: user could add/remove package in hob	
Steps: 1. launch hob 2. select one "Machine", for example, qemumips 3. choose one "Base image", for example, core-image-minimal 4. click "build image" button and wait for a successful build finished 4. click icon of "View Packages" and it will show a list of packages 5. select some un-selected packages and de-select some selected packages 6. check if the package number and dependency is correct	
Expected Results: user could add/remove package in hob	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2674: another build after one build is finished
Summary: User could start another build after one build is finished
Steps: 1. launch hob

2. select one "Machine", for example, qemuarm 3. choose one "Base image", for example, core-image-sato 4. click "build image" button and it should show a build progress bar 5. after build is finished, click "Build new image" 6. select another machine, for example, qemumips and choose another base image 7. click "build image" and wait for build finished	
Expected Results: User could start another build after one build is finished	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2675: My images shown	
Summary: User could check image list via "My images"	
Steps: 1. launch hob 2. select one "Machine", for example, qemuarm 3. choose one "Base image", for example, core-image-sato 4. click "build image" button and it should show a build progress bar 5. after build is finished, click "Build new image" 6. click "My images" and it should show a list of built out images	
Expected Results: User could check image list via "My images"	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2676: recipes added with new layer
Summary: Hob should be able to include new added recipes with new added layer
Steps: 1. launch hob

2. add a layer for meta-intel BSP, for example, add meta-intel/mata-emenlow and choose emenlow as Machine, choose "core-image-sato" for "Base image" 3. in "Settings", make sure "live" is selected for "Image types" 5. check if psb-firware is selected in "View Recipes", then click "Just bake" button and wait for a successful build finished 6. after build finished, choose "Deploy image" and insert a USB stick to burn the image into the stick 7. Use the stick to live boot on a real board and check if psb-firmware is installed	
Expected Results: Hob should be able to include new added recipes with new added layer	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2677: Extra paramters set in Others tab	
Summary: User could set extra parameters in "Settings"->"Others"	
Steps: 1. launch hob 2. click "Settings"->"Others", add extra paramters as following for libx11 #### PREFERRED_PROVIDER_virtual/libx11 = "libx11" #### 3. select one "Machine", for example, qemux86 4. choose one "Base image", for example, core-image-sato 5. build the recipe, libx11 and check if only libx11 is built out	
Expected Results: User could set extra parameters in "Settings"->"Others"	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2678: parameters remove in Others tab
Summary:

User could remove parameters in Others tab

Steps:

1. launch hob
2. click "Settings"->"Others", add extra paramters as following for libx11

####
PREFERRED_PROVIDER_virtual/libx11 = "libx11"
####

3. select one "Machine", for example, qemux86
4. check if libx11 is choosen for libx11 in "View Recipes", build a core-image-sato and check
5. back to "Others" and remove libx11
6. check if libx11 is removed for libx11 in "View Recipes", build a core-image-sato and check

Expected Results:

User could remove parameters in Others tab

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2679: Hob recipe build from bitbake command line

Summary:

User could build Hob recipes from bitbake command line

Steps:

1. launch hob
2. select one "Machine", for example, qemuarm
3. choose one "Base image", for example, core-image-sato
4. click "Just bake" button and it should build an image for you
5. exit Hob and copy bblayers-hob.conf and local-hob.conf into conf/ folder, replace the original bblayers.conf and local.conf
6. create an images folder named "recipes-test/images" under meta folder
7. run "bitbake hob-image" to build the hob recipe from bitbake command line

Expected Results:

User could build Hob recipes from bitbake command line

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

1.3 Test Suite : Distro

Test Case TC-2790: yocto build in Fedora 17	
Summary: Build latest yocto in x86_64 Fedora 17 host	
Steps: 1. By following the yocto handbook, download latest yocto source 2. Build core-image-minimal on Fedora 17	
Expected Results: Yocto build should pass on Fedora 17	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2791: yocto build in OpenSuse 12.1	
Summary: Build latest yocto in x86_64 OpenSuse 12.1	
Steps: 1. By following the yocto handbook, download latest yocto source 2. Build core-image-minimal on OpenSuse 12.1	
Expected Results: Build should pass on OpenSuse 12.1	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2792: yocto build in Ubuntu 12.04

Summary: Build latest yocto in x86_64 Ubuntu 12.04	
Steps: 1. By following the yocto handbook, download latest yocto source 2. Build core-image-minimal on Utuntu 12.04	
Expected Results: Yocto build should pass on Utuntu 12.04	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

1.4 Test Suite : System & Core OS

Test Case TC-2680: zypper command installed and workable	
Summary: check if zypper is installed and can work	
Steps: 1. Run command "zypper", and check the output	
Expected Results: Command "zypper" print the list of available global options and commands	
Test Execution Cycle Type:	Sanity
Case Automation Type:	Auto
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2681: zypper help search

Summary: check help option with zypper command	
Steps: 1. Run "zypper help search" and check the output	
Expected Results: The command should print help for the search command	
Test Execution Cycle Type:	Sanity
Case Automation Type:	Auto
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2682: zypper search package	
Summary: search package with zypper	
Steps: 1. Run "zypper search package_name" and check the output, for example "zypper search avahi"	
Expected Results: The command should search package "avahi" is installed or not	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Auto
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2683: zypper remove package
Summary: remove package with zypper
Steps:

1. Run "zypper rm pakcage_name" and check the output, for example "zypper rm avahi"
Expected Results: The command should remove package "avahi"

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2684: zypper install package
Summary: install package with zypper
Steps: 1. Set up a yum based repository on local server 2. Build out a package, which does not need any run-time dependency package, with local poky tree. For example, package "man" 3. In target system, run "zypper addrepo http://ip_address_of_repository zypper_test_repo" 4. Run "zypper refresh" to refresh the zypper repository cache 5. Run "zypper install package_name" and check the output, for example "zypper install man" to install package, which has no run-time dependency
Expected Results: The command should install package "man"

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2685: zypper install dependency package

Summary: install dependency package with zypper
Steps: 1. Set up a yum based repository on local server 2. Build out a package, which does not need any run-time dependency package, with local poky tree. For example, package "mc" 3. In target system, run "zypper addrepo http://ip_address_of_repository zypper_test_repo" 4. Run "zypper refresh" to refresh the zypper repository cache 5. Run "zypper install package_name" and check the output, for example "zypper install mc" to install package, which needs run-time dependency packages installed also, like ncurses-terminfo.
Expected Results: The command should install package "mc" and denpendency package ncurses-terminfo.

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2686: zypper install .all packages
Summary: install packages from all folder with zypper
Steps: 1. Set up a yum based repository on local server 2. Build out a package, which belongs to all folder, for example, xcursor-transparent-theme-dbg-0.1.1-r3.all.rpm. 3. In target system, run "zypper addrepo http://ip_address_of_repository zypper_test_repo" 4. Run "zypper refresh" to refresh the zypper repository cache 5. Run "zypper install xcursor-transparent-theme-dbg" and check the output
Expected Results: package install from all folder should be installed successfully with zypper

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver

image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2687: rpm query package	
Summary: make sure rootfs image is built with rpm packages	
Steps: 1. launch terminal 2. run command "rpm -qa", which lists all existing packages in system	
Expected Results: "rpm -qa" should print all existing packages in system	
Test Execution Cycle Type:	Sanity
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2688: rpm install package	
Summary: rpm format package can be installed	
Steps: 1. Get a RPM package(for example, man) from zypper repository or build one on local machine 2. Copy the package into image, run command "rpm -ivh package_name" to install the package	
Expected Results: RPM format package can be installed	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk

Last Result	**Not Run**
Keywords:	None

Test Case TC-2689: rpm install dependency package	
Summary: rpm command should report dependency when installing package	
Steps: 1. Get a RPM package or build one on local machine, which should have run-time dependency. For example, mc should depend on ncurses-terminfo 2. Run "rpm -ivh package_name" and check the output, for example "rpm -ivh mc.rpm*" should report the dependency on ncurses-terminfo	
Expected Results: rpm command should report message when some RPM installation depends on other packages	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2690: rpm remove package	
Summary: rpm command can remove package in system	
Steps: 1. Launch terminal and run command "rpm -e package_name" to remove some package, for example, avahi	
Expected Results: RPM package can be removed by command rpm	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**

Keywords:	None

Test Case TC-2691: check rpm install/removal log file size	
Summary: The case is to track log file size after rpm install/removal	
Steps: 1. After system is up, check the log file size after rpm/zypper install/removal 2. for rpm, there will be some database files under /var/lib/rpm/, named as "__db.xxx" and there will be some log files under /var/lib/rpm/log, named as "log.xxxxxx". Each file will occupy about 10MB. 3. after several rpm/zypper install/removal, rpm will create several log files under /var/lib/rpm/log, which eat lots of system disk space.	
Expected Results: there should be some method to keep rpm log in a small size	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2692: boot and install from USB	
Summary: boot and install image from usb stick	
Steps: 1. plugin usb which contains live image burned 2. configure device BIOS to firstly boot from USB if necessary 3. boot the device and select some option like "Boot and Install" from boot menu 4. proceed through default install process 5. Remove USB, and reboot into new installed system.	
Expected Results: 1. User can choose install system from usb stick onto harddisk from boot menu or command line option 2. Imstalled system can boot up	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	undecided
target:	e-menlow, blacksand, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2693: live boot from USB	
Summary: live boot from USB	
Steps: boot live image from usb stick 1. plugin usb which contains live image burned 2. configure device BIOS to firstly boot from USB if necessary 3. reboot the device and boot from USB stick	
Expected Results: 1. User can choose boot from live image on usb stick from boot menu or command line option 2. Live image can boot up with usb stick	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	undecided
target:	e-menlow, blacksand, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2694: boot from runlevel 3	
Summary: Verify that system can boot from runlevel 3	
Steps: 1. Boot into system and edit /etc/inittab to make sure system enter init 3 by default ######## id:3:initdefault ######## 2. reboot system, and press Tab to enter "grub" 3. edit "kernel" line and add "psplash=false text" at the end 4. Press "F10" or "ctrl+x" to boot system	
Expected Results: system should boot to runlevel 3.	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	undecided
target:	e-menlow, blacksand, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk

Last Result	**Not Run**
Keywords:	None

Test Case TC-2695: boot from runlevel 5	
Summary: Verify that system can boot from runlevel 5	
Steps: 1. Boot into system and edit /etc/inittab to make sure system enter init 5 by default ######## id:5:initdefault ######## 2. reboot system, and press Tab to enter "grub" 3. edit "kernel" line and make sure no "psplash=false text" in grub cmdline 4. Press "F10" or "ctrl+x" to boot system Note: The test is only for sato image.	
Expected Results: system should boot to runlevel 5.	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	undecided
target:	e-menlow, blacksand, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2696: g++ compile in sdk image
Summary: check if g++ can compile program in sdk image
Steps: 1. Boot up sdk image 2. check if g++ is built in 3. compile following program test.c "g++ test.c -o test -lm" 4. run "./test" and check the output is correct   test.c: ########## #include <stdio.h> #include <math.h>

```
double
convert(long long l)
{
  return (double)l;    // or double(l)
}

int
main(int argc, char * argv[])
{
  long long l = 10;
  double f;

  f = convert(l);
  printf("convert: %lld => %f\n", l, f);

  f = 1234.67;
  printf("floorf(%f) = %f\n", f, floorf(f));
  return 0;
}
##########
```

Expected Results:

executable binary test can run without problem

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2858: syslogd workable

Summary:

Check if syslogd can work.

Steps:

1. boot system
2. run "ps aux | grep syslogd" or "ps -ef | grep syslogd"
3. check if there is a process named syslogd in background
4. run “cat /var/log/message”

Expected Results:

There should be a process named syslogd in background. The log message should be recorded to /var/log/message .

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest,

	FRI2, HuronRiver
Last Result	**Not Run**
Keywords:	None

Test Case TC-2697: syslog configurable	
Summary: Check if syslog could be configured by user and run without problem	
Steps: 1. Get a yocto image from autobuilder or local build 2. Launch image and check if syslog is started by default in background with ps command 3. Modify /etc/syslog-startup.conf, change the LOGFILE to /var/log/messages.test 4. Restart syslog with command "/etc/init.d/syslog restart" 5. Check if syslog is started in background with ps command 6. Check if there is file generated under /var/log/messages.test	
Expected Results: syslog could be configured by user and run without problem	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2698: gcc compile in sdk image
Summary: check if gcc can compile program in sdk image

Steps:
 	
1. Boot up sdk image
2. check if gcc is built in
3. compile following program test.c "gcc test.c -o test -lm"
4. run "./test" and check the output is correct
 	
test.c:
##########
#include <stdio.h>
#include <math.h>
 	
double
convert(long long l)
{
  return (double)l; // or double(l)
}
 	
int

```
main(int argc, char * argv[])
{
  long long l = 10;
  double f;

  f = convert(l);
  printf("convert: %lld => %f\n", l, f);

  f = 1234.67;
  printf("floorf(%f) = %f\n", f, floorf(f));
  return 0;
}
##########
```

Expected Results:

executable binary test can run without problem

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2699: run command make in sdk image

Summary:

check if command make can work in sdk image

Steps:

1. Boot up sdk image
2. check if make is built in
3. run command "make" with following makefile and build the test.c file from case "gcc compile in sdk image"

```
test: test.o
	gcc -o test test.o -lm
test.o: test.c
	gcc -c test.c
```

Expected Results:

make command can work without problem

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver

image profile:	sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2700: cvs project compile in sdk image	
Summary: cvs project could be compiled in sdk image	
Steps: 1. Download cvs project from http://ftp.gnu.org/non-gnu/cvs/source/feature/1.12.13/cvs-1.12.13.tar.bz2 2. Copy cvs tarball into sdk image 3. Extract the tarball and do "configure", "make" and "make install"	
Expected Results: cvs project could be compiled successfully	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2701: iptables project compile in sdk image	
Summary: iptables project could be compiled in sdk image	
Steps: 1. Download iptables project from http://netfilter.org/projects/iptables/files/iptables-1.4.11.tar.bz2 2. Copy iptables tarball into sdk image 3. Extract the tarball and do "configure", "make" and "make install"	
Expected Results: iptables could be compiled successfully	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato-sdk, lsb-sdk

Last Result	**Not Run**
Keywords:	None

Test Case TC-2702: sudoku-savant project compile in sdk image	
Summary: sudoku-savant could be compiled in sdk image	
Steps: 1. Download sudoku-savant project from http://downloads.sourceforge.net/project/sudoku-savant/sudoku-savant/sudoku-savant-1.3/sudoku-savant-1.3.tar.bz2 2. Copy sudoku-savant tarball into sdk image 3. Extract the tarball and do "configure", "make"	
Expected Results: sudoku-savant could be compiled successfully	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2703: perl program work in image	
Summary: A perl program could be executed and output correctly in image	
Steps: 1. Check if perl is installed in image and could run with "perl -v" 2. Prepare a perl program like followig test.pl 3. Run "perl test.pl" ######## $a = 9.01e+21 + 0.01 - 9.01e+21; print ("the value of a is ", $a, "\n"); $a = 9.01e+21 - 9.01e+21 + 0.01; print ("the value of a is ", $a, "\n"); ########	
Expected Results: The test.pl could run without problem	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Auto

Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2704: shutdown system	
Summary: verify that system can be shutdown by command	
Steps: 1. boot system 2. launch terminal and run "shutdown -h now" or "poweroff"	
Expected Results: System can be shutdown successfully	
Test Execution Cycle Type:	Sanity
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, mpc8315e-rdb, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2705: reboot system	
Summary: verify that system can boot by command	
Steps: 1. boot system 2. launch terminal and run "reboot"	
Expected Results: System can reboot successfully	
Test Execution Cycle Type:	Sanity
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver

image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2706: adjust date and time	
Summary: adjust date and time	

Steps:
 	
1.launch terminal and run "date -R" to check current system time
2.adjust Date&Time by these commands:
For date command from coreutils, for example the sdk image use coreutils, you should use following syntax:
$ date -s "10:00:00 20100809"
$ date -R
$ Mon, 09 Aug 2010 10:00:00 +0000
For date command in busybox, for example the sato image use busybox, you should use following syntax:
$ date "080910002010"
$ date -R
$ Mon, 09 Aug 2010 10:00:00 +0000
3. check date with "date -R" and the time shown on matchbox-panel	

Expected Results: System time should be adjust to what you specified	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Auto
Case State:	Ready
Feature:	system usage
target:	e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2707: switch among multi applications and desktop	
Summary: switch among multi applications and desktop	
Steps: 1. launch several applications(like contacts, file manager) 2. launch terminal 3. switch among multi applications and desktop 4. close applications Note: The case is for sato image only.	
Expected Results: 1. user could switch among multi applications and desktop	

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	e-menlow, blacksand, beagleboard, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2708: vncserver for target	
Summary: Check if vncserver setup work in target and vnc client could connect it	
Steps: 1. Check if x11vnc is installed in target by running "which x11vnc" 2. Run command "x11vnc -display :0.0", check the ip address of the target 3. On a client, run command "vncviewer $ip_address_of_target:0"	
Expected Results: A virtual X desktop of target should be pop-up on the client	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2709: file manager	
Summary: file manager	
Steps: 1.launch file manager from application panel 2.view folder/file in file manager 3.copy and paste folder/file in file manager Note: The test is only for sato image	
Expected Results: 1.folder and file could be listed in file browser with different display mode	
Test Execution Cycle Type:	Weekly

Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	e-menlow, blacksand, beagleboard, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2710: system dmesg log check	
Summary: check if there is error in dmesg after system boot up	
Steps: 1.make sure no other operation after stattup the system. 2.run "dmesg | grep -i error" in the terminal. 3.check if there is any error log printed.	
Expected Results: No error message in dmesg	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2711: usb mount	
Summary: verify that system can mount plugged usb automatically	
Steps: 1. boot system 2. plug usb stick	
Expected Results: 1. system notify that usb stick is accessible	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage

target:	e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2712: usb read files	
Summary: verify that system can read files from usb	
Steps: 1. boot system 2. plug usb stick 3. view files in usb by file browser 4.copy some files from usb to local hardware	
Expected Results: 1. view/copy successfully	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2713: usb umount	
Summary: verify that system can unmout usb automically	
Steps: 1. boot system 2. plug usb stick 3. view files in usb by file browser 4.unplug usb	
Expected Results: 1. usb direcoty in file browser automatically missed	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver

image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2714: usb write files	
Summary: verify that system can write files to usb	
Steps: 1. boot system 2. plug usb stick 3. create files in usb 4.copy some files from local hardware to usb	
Expected Results: 1. create/copy successfully	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2715: file copy by scp	
Summary: check if file can be copied from remote machine to device by scp	
Steps: 1. check avahi is install and started 2. get system IP and try "scp file $IP:/home/root" from remote machine (file >= 500M for real HW, file>=5M for QEMU)	
Expected Results: File can be copied from remote machine to device by scp	
Test Execution Cycle Type:	Sanity
Case Automation Type:	Auto
Case State:	Ready
Feature:	connectivity
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**

Keywords:	None

Test Case TC-2716: connman launch after boot	
Summary: After system booted, the connmand daemon should be launched	
Steps: 1. boot system 2. "ps aux | grep connmand" or "ps -ef | grep connmand" 3. check if there is a thread named connmand in background	
Expected Results: There should be one thread named connmand in background	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	connectivity
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2717: ethernet enabled in connman	
Summary: After system boot, ethernet can get IP address with connman	
Steps: 1. boot system with network cable plugged in 2. "ps aux |grep connmand" or ""ps -ef | grep connmand" to check if connmand is started 3. "ifconfig" check ethernet could get IP address and ping the address from remote machine	
Expected Results: Ethernet interface can get IP via connman	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	connectivity
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2718: only one connmand in background	
Summary: there should be no more than one connmand in background	
Steps: 1. boot system 2. "ps aux |grep connmand" or "ps -ef | grep connmand" 3. the connmand should be in background 4. run command "connmand" 5. check if the second connmand can be generated	
Expected Results: There will be only one connmand instance in background	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	connectivity
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2719: remote access by ssh	
Summary: check if the device can be accessed remotely by ssh	
Steps: 1. check dropbear is install and started 2. get system IP and try "ssh $IP" from remote machine	
Expected Results: it is ok to access system by ssh from remote machine	
Test Execution Cycle Type:	Sanity
Case Automation Type:	Auto
Case State:	Ready
Feature:	connectivity
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2720: ethernet static ip set in connman	
Summary: we could set static ip for ethernet in connman	
Steps: 1. launch connman-properities 2. choose ethernet device and set static ip for it. For example, in our internal network, we can set as following: ip address: 10.239.48.xxx Broadcast: 10.239.48.255 Mask: 255.255.255.0	
Expected Results: we can set static ip for ethernet device	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	connectivity
target:	e-menlow, blacksand, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2721: ethernet get IP in connman via DHCP	
Summary: ethernet device can get IP in connman via DHCP	
Steps: 1. Set static IP for ethernet device in connman 2. Check if ethernet device can work with static IP 3. Choose DHCP method for ethernet device 4. Check with ping if ethernet device get IP address via DHCP	
Expected Results: Ethernet device can get dynamic IP address via DHCP in connman	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	connectivity
target:	e-menlow, blacksand, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2722: connman offline mode in connman-gnome	
Summary: change offline mode in comman-gnome can make all connection off	
Steps: 1. Launch connman-properties after system booting 2. choose "offline mode" and check the connection of all network interfaces	
Expected Results: All connection should be off after clicking "offline mode"	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	connectivity
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2723: X server can start up with runlevel 5 boot	
Summary: check if X server can work well after system runlevel 5 booting	
Steps: 1. boot up system with default runlevel	
Expected Results: X server can start up well and desktop display has no problem	
Test Execution Cycle Type:	Sanity
Case Automation Type:	Auto
Case State:	Ready
Feature:	graphics
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, beagleboard, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2724: qt application quicky	
Summary:	

quicky is a simple note-taking application with Wiki-style syntax and behaviour

Steps:

launch quicky and write something in quicky

Expected Results:

http://qt-apps.org/content/show.php/Quicky?content=80325

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	graphics
target:	e-menlow, blacksand, beagleboard, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2725: standby

Summary:

system can enter standby and resume from standby

Steps:

1. boot system and launch terminal; check output of "date" and launch script "continue.sh"
2. echo "mem" > /sys/power/state
3. After system go into S3 mode, move mouse or press any key to make it resume
4. Check "date" and script "continue.sh"
5. Check if application in X can work as normal

continue.sh as below:

```
##################
#!/bin/sh

i=1
while [ 0 ]
do
 echo $i
 sleep 1
 i=$((i+1))
done
##################
```

Expected Results:

screen should resume back and script can run continuously

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	e-menlow, blacksand, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato-sdk

Last Result	**Not Run**
Keywords:	None

Test Case TC-2726: check CPU utilization after standby	
Summary: check CPU utilization after standby	
Steps: 1. Start up system 2. run "top" command and check if there is any process eating CPU time 3. make system into standby and resume it 4. run "top" command and check if there is any difference with the data before standby	
Expected Results: There should be no big difference before/after standby with "top"	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	crownbay, sugarbay, FRI2, HuronRiver
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2727: Test if LAN device works well after resume from suspend state	
Summary: Test if LAN device works well after resume from suspend state.	
Steps: 1. boot system and launch terminal 2. echo "mem" > /sys/power/state 3. After system go into S3 mode, move mouse or press any key to make it resume 4. check ping status	
Expected Results: ping should always work before/after standby	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	e-menlow, blacksand, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2728: Test if usb hid device works well after resume from suspend state	
Summary: Test if usb hid device works well after resume from suspend state.	
Steps: 1. boot system and launch terminal 2. echo "mem" > /sys/power/state 3. After system go into S3 mode, move mouse or press any key to make it resume 4. check usb mouse and keyboard	
Expected Results: usb mouse and keyboard should work	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	e-menlow, blacksand, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2729: disk space check	
Summary: There should be enough disk space for QEMU rootfs	
Steps: 1. Launch QEMU targets(with rootfs.ext3 file) 2. Check the output of command df 3. If there is less than 5M disk space available, we assume it a failure	
Expected Results: There should be enough disk space for QEMU targets	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2730: click terminal icon on X desktop
Summary: terminal icon should work without problem on X desktop

Steps: 1. After system launch and X start up, click terminal icon on desktop 2. Check if only one terminal window launched and no other problem met	
Expected Results: there should be no problem after launching terminal	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, beagleboard, crownbay, sugarbay, FRI2, HuronRiver
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2731: Add multiple files in music player	
Summary: music player should be no problem when adding multiple files at same time	
Steps: 1. Launch music player 2. Add multiple files(5 files) in music player at same time	
Expected Results: music player should be OK with this action	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	e-menlow, blacksand, beagleboard, crownbay, sugarbay, FRI2, HuronRiver
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2732: system shutdown with UNFS
Summary: system shutdown with UNFS should work
Steps: 1. Use UNFS to start QEMU targets 2. Run shutdown in QEMU targets
Expected Results:

QEMU shutdown with UNFS should work	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	qemux86_32, qemux86_64, qemuarm, qemumips
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2733: no connman-gnome icon on desktop	
Summary: there should be no connman-gnome icon on desktop	
Steps: 1. Launch sato image 2. There should be no connman-gnome icon on desktop, and connman-properties should be only invoked by toolbar	
Expected Results: There should be no connman-gnome icon on desktop, and connman-properties should be only invoked by toolbar	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, beagleboard, crownbay, sugarbay, FRI2, HuronRiver
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2734: application contacts should work	
Summary: application contacts should work without problem	
Steps: 1. Make sure X is started up 2. Check if there is "contacts" icon on desktop and run it 3. Check if there is any error by checking the output of this action and dmesg log	
Expected Results: "contacts" launch should not cause any error	
Test Execution Cycle Type:	Weekly

Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, beagleboard, crownbay, sugarbay, FRI2, HuronRiver
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2735: x11vnc icon click for target	
Summary: Check if vncserver could work in target by clicking x11vnc icon	
Steps: 1. Check if there is a x11vnc icon in target 2. Click the x11vnc icon and check the ip address of the target 3. On a client, run command "vncviewer $ip_address_of_target:0"	
Expected Results: A virtual X desktop of target should be pop-up on the client	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, crownbay, sugarbay, FRI2, HuronRiver
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2736: RTLDLIST path check for ldd command	
Summary: check if the file set in RTLDLIST is valid	
Steps: 1. After system is up, check if the RTLDLIST variable in ldd command 2. The file path set in RTLDLIST should be valid	
Expected Results: check if the file set in RTLDLIST is valid	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage

target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2737: check bash in image	
Summary: check if bash exists in image	
Steps: 1. After system is up, check if bash command exists with command "which bash"	
Expected Results: bash command should exist in image	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips, e-menlow, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest, FRI2, HuronRiver
image profile:	sato, sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2738: "Install/Remove Software" icon should be removed	
Summary: "Install/Remove Software" icon should be removed from sato	
Steps: 1. After system is up, there should be no "Install/Remove Software" icon	
Expected Results: "Install/Remove Software" icon should be removed	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	qemux86_32, qemux86_64, qemuarm, qemuppc, qemumips
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2843: MicroSD mount	
Summary: verify that system can mount plugged MicroSD card automatically	
Steps: 1. boot system 2. plug MicroSD card	
Expected Results: 1. system notify that MicroSD is accessible	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	FRI2
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2844: MicroSD read files	
Summary: verify that system can read files from MicroSD	
Steps: 1. boot system 2. plug MicroSD card 3. view files in MicroSD by file browser 4.copy some files from MicroSD to local hardware	
Expected Results: 1. view/copy successfully	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	FRI2
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2845: MicroSD umount
Summary:

verify that system can unmout MicroSD card automically
Steps: 1. boot system 2. plug MicroSD card 3. view files in MicroSD by file browser 4.unplug MicroSD
Expected Results: 1. MicroSD in file browser automatically missed

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	FRI2
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2846: MicroSD write files
Summary: verify that system can write files to MicroSD
Steps: 1. boot system 2. plug MicroSD card 3. create files in MicroSD 4.copy some files from local hardware to MicroSD
Expected Results: 1. create/copy successfully

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	FRI2
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2847: eSATA support
Summary: check if eSATA AHCI support by system
Steps:

1. install one eSata disk into the system, enter system BIOS, configure system boot from Sata 2. reboot the system and boot up yocto 3. check if yocto system could boot up and harddisk could read/wrote without problem
Expected Results: check if eSATA AHCI mode could work well

Test Execution Cycle Type:	BAT
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	FRI2
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2848: wifi - wifi automatic refersh
Summary: check if connman could get AP automatically
Steps: 1. Prepare a WIFI AP, setting it as SSID broadcast. 2. Make sure ConnMan is launched, and list all networks, check if the WIFI AP can be scaned.
Expected Results: The WIFI AP can be scanned automatically in the ConnMan

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	connectivity
target:	FRI2
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2849: wifi - ethernet and wifi coexist
Summary: check if ethernet and WIFI can coexists when connect wired network fist
Steps: 1. Connect a wired network first 2. Connect a WIFI network then 3. Check default gateway 4. Ping the 2 DHCP server of the wired and WIFI networks.
Expected Results:

Default gateway should be of the wired network and can ping both the DHCP servers.

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	connectivity
target:	FRI2
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2850: wifi - wifi and ethernet coexist
Summary: check if wired and WIFI can coexists when connect WIFI network fist
Steps: 1. Connect a WIFI network first 2. Connect a wired network then 3. Check default gateway 4. Ping the 2 DHCP server of the wired and WIFI networks.
Expected Results: Default gateway should be of the WIFI network and can ping both the DHCP servers.

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	connectivity
target:	FRI2
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2851: wifi - connect to AP
Summary: check if user could connect to AP in comman
Steps: Pre-condition: the connman daemon is runnig wifi device is enabled. 1. make sure there is at least a AP running and could be scanned 2. open connman and check the AP in connection list 3. connect the AP and check the IP address with "ifconfig wlan0" 4. send ping packets to AP, check if can receive ping reply from AP.
Expected Results: ConnMan shall connect with this AP and get available IP address.

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	connectivity
target:	FRI2
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2852: wifi - disconnect from AP	
Summary: check if user could disconnect from AP in comman	
Steps: Pre-condition: the connman daemon is runnig wifi device is enabled. 1. connect to AP in connman and make sure it work well 2. in connman, disconnect the AP 3. check IP address with "ifconfig wlan0", it should have no IP address	
Expected Results: user could disconnect from AP and there should be no IP address assigned to wlan0	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	connectivity
target:	FRI2
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2853: wifi - file copy by scp
Summary: check if file can be copied from remote machine to device by scp with wifi
Steps: Pre-condition: the connman daemon is runnig wifi device is enabled. 1. connect to AP in connman and make sure wlan0 could get IP address 2. get system IP and try "scp file $IP:/home/root" from remote machine (file >= 500M for real HW)
Expected Results: File can be copied from remote machine to device by scp via wifi

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	connectivity
target:	FRI2
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2854: wifi - Test if wifi device works well after resume from suspend state	
Summary: Test if wifi device works well after resume from suspend state.	
Steps: Pre-condition: the connman daemon is runnig wifi device is enabled. 1. connect to AP in connman and make sure wlan0 could get IP address 2. echo "mem" > /sys/power/state 3. After system go into S3 mode, move mouse or press any key to make it resume 4. check wlan0 connection status	
Expected Results: wifi interface should always work before/after standby	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	FRI2
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2855: 3G - get IP address	
Summary: user could get IP address via 3G interface	
Steps: 1. boot up system and launch connman 2. in "cellular networks", there should be an interface for 3G connection, like "china unicom" 3. connect to the 3G point and check ip address with ifconfig	
Expected Results: user could get IP address via 3G interface	
Test Execution Cycle Type:	Weekly

Case Automation Type:	Manual
Case State:	Ready
Feature:	connectivity
target:	FRI2
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2856: 3G - ping public website	
Summary: user could ping public website with 3G connection	
Steps: 1. boot up system and launch connman 2. in "cellular networks", there should be an interface for 3G connection, like "china unicom" 3. connect to the 3G point and check ip address with ifconfig 4. ping some public website, like www.baidu.com, and check the output	
Expected Results: user could ping public website with 3G connection	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	connectivity
target:	FRI2
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2857: 3G - disconnect from 3G point	
Summary: user could disconnect from 3G connection in connman	
Steps: 1. boot up system and launch connman 2. in "cellular networks", there should be an interface for 3G connection, like "china unicom" 3. connect to the 3G point and check ip address with ifconfig 4. click disconnect from the 3G point in connman and check network status with ifconfig	
Expected Results: user could disconnect from 3G connection in connman	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	connectivity

target:	FRI2
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

1.5 Test Suite : Stress

Test Case TC-2739: crashme for stress	
Summary: Run crashme in real hardware for stress testing	
Steps: 1. Get crashme from http://people.delphiforums.com/gjc/crashme.html 2. By following the setup steps on above URL, build crashme in target. 3. Run crashme for 24 hours	
Expected Results: target should not crash with the program	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	stress
target:	beagleboard, jasperforest
image profile:	sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2740: helltest for stress	
Summary: Run helltest for stress in target	
Steps: 1. helltest is stress test suite, which does compiler test for hours 2. We download the test suite and run it for 24 hours	
Expected Results: helltest should not make target crash	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	stress
target:	jasperforest

image profile:	lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2741: ltp for stress	
Summary: Run ltp stress in real hardware for stress testing	
Steps: LTP download: ./meta/recipes-extended/ltp/ltp_20120401.bb build steps: refer to http://ltp.sourceforge.net Run steps: 1. Build LTP with toolchain or in sdk image 2. Copy LTP folder into target, for example, /opt/ltp. Modify script "testscripts/ltpstress.sh", set “lostat=1”, ”NO_NETWORK=1” 3. cd testscripts/ && ./ltpstress.sh 4. This stress case will run for 24 hours	
Expected Results: Check the result, target should not crash with the program.	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	stress
target:	beagleboard
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

1.6 Test Suite : Power/Performance

Test Case TC-2742: boot time collection
Summary: To collect boot time of clean installation, from grub to full desktop
Steps: 1. Reboot testing device at least 3 times and do not plug anything while collecting boot time by stopwatcher: #reboot
Expected Results:

Provide average boot time and dmesg log

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	performance
target:	crownbay
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2743: memory footprint
Summary: collect data of the used/free memory
Steps: With default installtion, launch terminal and type 'free' to read the used/free disk space
Expected Results: Provide 'free' output

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	core
target:	crownbay
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2744: powertop log
Summary: collect powertop data
Steps: 1. Run "powertop -d" and record output 2. Save the percentage of deepest C state(C3 or C2)
Expected Results: Provide powertop output

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready

Feature:	core
target:	crownbay
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2745: Idle power consumption	
Summary: Collect idle power consumption of target system	
Steps: 1. Use power meter to collect ilde power consumption of target system for 10 minutes 2. Save it and compare it with old data	
Expected Results: There should be no regression between old and new ilde power data	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	performance
target:	crownbay
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2746: core build time for sato image	
Summary: collect the core build time for sato qemux86 image	
Steps: 1. Perpare a system with following configuration CPU: 4-core * 2-threads Intel(R) Core(TM) i7 CPU 860 @ 2.80GHz Memory: 4GB Harddisk: 1TB OS: Ubuntu 10.04 x86_64 Kernel: 2.6.32-21 2. Download poky tree and make sure all the source packages have been downloaded 3. Build a qemux86 sato image and collect the time	
Expected Results: There should be no regression for build time	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual

Case State:	Ready
Feature:	performance
target:	qemux86_32
image profile:	sato
Last Result	**Not Run**
Keywords:	None

1.7 Test Suite : Graphics

Test Case TC-2747: Graphics ABAT

Summary:

Yocto on SugarBay should pass Intel graphics ABAT testing

Steps:

1. Download ABAT test suite from internal git repository, git clone git://tinderbox.sh.intel.com/git/abat
2. Apply following patch to make it work on yocto environment
3. Run "./abat.sh" to run ABAT test

########

```
diff --git a/glxgears_check.sh b/glxgears_check.sh
index 17622b8..c4d3b97 100755
--- a/glxgears_check.sh
+++ b/glxgears_check.sh
@@ -31,7 +31,7 @@ else

     sleep 6

-    XPID=$( ps ax | awk '{print $1, $5}' | grep glxgears | awk '{print $1}')
+    XPID=$( ps | awk '{print $1, $5}' | grep glxgears | awk '{print $1}')
     if [ ! -z "$XPID" ]; then
         kill -9 $XPID >/dev/null 2>&1
         echo "glxgears can run, PASS!"
diff --git a/x_close.sh b/x_close.sh
index e287be1..3429f1a 100755
--- a/x_close.sh
+++ b/x_close.sh
@@ -22,7 +22,7 @@
 #
 function close_proc(){
 echo "kill process Xorg"
-XPID=$(  ps ax | awk '{print $1, $5}' | egrep "X$|Xorg$" | awk '{print $1}')
+XPID=$(  ps | awk '{print $1, $6}' | egrep "X$|Xorg$" | awk '{print $1}')
 if [ ! -z "$XPID" ]; then
     kill $XPID
     sleep 4
diff --git a/x_start.sh b/x_start.sh
index 9cf6eab..2305796 100755
--- a/x_start.sh
+++ b/x_start.sh
@@ -24,7 +24,7 @@
 X_ERROR=0

 #test whether X has started
-PXID=$(ps ax |awk '{print $1,$5}' |egrep "Xorg$|X$" |grep -v grep | awk '{print $1}')
+PXID=$(ps |awk '{print $1,$6}' |egrep "Xorg$|X$" |grep -v grep | awk '{print $1}')
 if [ ! -z "$PXID" ]; then
```

```
    echo "[WARNING] Xorg has started!"
    XORG_STATUS="started"
@@ -35,9 +35,11 @@ else
    #start up the x server
    echo "Start up the X server for test in display $DISPLAY....."

-    $XORG_DIR/bin/X >/dev/null 2>&1 &
+    #$XORG_DIR/bin/X >/dev/null 2>&1 &
+    #sleep 8
+    #xterm &
+    /etc/init.d/xserver-nodm start &
    sleep 8
-    xterm &
 fi
    XLOG_FILE=/var/log/Xorg.0.log
    [ -f $XORG_DIR/var/log/Xorg.0.log ] && XLOG_FILE=$XORG_DIR/var/log/Xorg.0.log
@@ -54,7 +56,7 @@ fi
        X_ERROR=1
    fi

-    XPID=$( ps ax | awk '{print $1, $5}' | egrep "X$|Xorg$" |grep -v grep| awk '{print $1}')
+    XPID=$( ps | awk '{print $1, $6}' | egrep "X$|Xorg$" |grep -v grep| awk '{print $1}')
    if [ -z "$XPID" ]; then
        echo "Start up X server FAIL!"
  echo
########
```

Expected Results:

All ABAT test should pass

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	bsp
target:	e-menlow, blacksand, crownbay, sugarbay, FRI2, HuronRiver
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2748: openarena - 3D

Summary:

Run opernarena testing and compare the result with upstream graphics result

Steps:

1. Download and build openarena through phoronix test suite. first download a new phoronix from its website, then download the game in it. The openarena we use is v0.8.5.
####
phoronix-test-suite list-tests
phoronix-test-suite install openarena
####
2.Go into the directory of openarena sourcecode folder.
3.Find the correct name of ld-linux.so needed by openarena, for example, it should be "/lib64/ld-linux-x86-64.so.2" in the openarena.x86_64 if you grep it.
4.Check if /lib64/ld-linux-x86-64.so.2 exists on system. If not, we need to create a link file linking to the real path of ld-linux in system. For example, on a x86_64 machine, the commands should be "mkdir /lib64 && ln -s /lib/ld-linux-x86-64.so.2 /lib64/ld-linux-x86-64.so.2".
5.Modify the path to make sure the openareana can find the correct executable file, openarena.i386 for x86 host and openarena.x86_64 for x86_64 host.
6.Run the test suite with following command:
vblank_mode=0 ./openarena +exec pts +set r_mode -1 +set r_fullscreen 1 +set r_customWidth

$VIDEO_WIDTH +set r_customHeight $VIDEO_HEIGHT The VIDEO_WIDTH and VIDEO_HEIGHT set the game's resolution，you can get current resolution by command "xrandr"	
Expected Results: Compare the result of Yocto with upstream graphics	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	bsp
target:	sugarbay, HuronRiver
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2749: urbanterror - 3D	
Summary: Run urbanterror and compare the result of Yocto with upstream graphics	
Steps: 1. Download and build: This game also can get through phoronix-test-suite. 2.We should modify script urbanterror by setting following options before test: ### OS_TYPE=Linux OS_ARCH=`uname -m` LOG_FILE=/home/root/log ### 3. touch a log file /home/root/log 3. Run urbanterror with following command ### vblank_mode=0 ./urbanterror +timedemo 1 +set demodone 'quit' +set demoloop1 'demo pts1; set nextdemo vstr demodone' +vstr demoloop1 +set r_customwidth $VIDEO_WIDTH +set r_customheight $VIDEO_HEIGHT ###	
Expected Results: Get the FPS data of Yocto and compare it with upstream graphics	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	bsp
target:	sugarbay, HuronRiver
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2750: x11perf - 2D
Summary:

Get fps data of x11per running
Steps: 1. Run "x11perf -aa10text" and "x11perf -rgb10text" 2. Get the FPS result and compare it with upstream graphics data on Sandybridge
Expected Results: There should not be big regression between Yocto and upstream linux

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	bsp
target:	sugarbay, HuronRiver
image profile:	sato, sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2841: xorg.conf check
Summary: check if xorg.conf set emgd as default driver
Steps: 1. after system is launched, check /etc/X11/xorg.conf, device driver should be set to "emgd" in Section "Device"
Expected Results: emgd driver should be set in xorg.conf

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	graphics
target:	FRI2
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

1.8 Test Suite : Mulitimedia

Test Case TC-2751: libva check (ogg video play)
Summary:

check if libva is installed and used when video player playing ogg video file
Steps: 1. check if libva is installed on system 2. copy sample ogg file to system 3. launch video player can play the ogg file
Expected Results: ogg file can be played without problem when libva is used

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	multi-media
target:	e-menlow, blacksand, crownbay, sugarbay
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2752: sound on/off
Summary: check if sound can be turned on/off
Steps: 1. copy amixer is installed 2. Run "amixer set Master on" to turn on audio device 3. Run "amixer set Master 64" to adjust to maxium volumn 4. Run "amixer set Speaker on" to turn on speaker 5. Run "amixer set Speaker 64" to adjust to maxium volumn 6. Run "amixer set Master off" to turn off audio device 7. Run "amixer set Speaker off" to turn off speaker
Expected Results: Above commands can run without problem

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	multi-media
target:	e-menlow, blacksand, beagleboard, crownbay, sugarbay
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2753: audio play (mp3)
Summary: make sure music player cannot play mp3 format file

Steps: 1. copy sample mp3 file to system 2. launch music player and make sure it cannot play the mp3 file	
Expected Results: mp3 file can not be played	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	multi-media
target:	e-menlow, blacksand, beagleboard, crownbay, sugarbay
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2754: audio play (ogg)	
Summary: check if music player can play ogg format file	
Steps: 1. copy sample ogg file to system 2. launch music player can play the ogg file	
Expected Results: ogg file can be played without problem	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	multi-media
target:	e-menlow, blacksand, beagleboard, crownbay, sugarbay
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2755: audio stop (ogg)
Summary: check if music player can play ogg format file
Steps: 1. copy sample ogg file to system 2. launch music player can play the ogg file 3. click "stop" button to stop playing 4. click "start" button to resume playing
Expected Results:

ogg file can be start/stop without problem

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	multi-media
target:	e-menlow, blacksand, beagleboard, crownbay, sugarbay
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2756: audio play (wav)
Summary: check if music player can play wav format file
Steps: 1. copy sample wav file to system 2. launch music player can play the wav file
Expected Results: wav file can be played without problem

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	multi-media
target:	e-menlow, blacksand, beagleboard, crownbay, sugarbay
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2757: audio stop (wav)
Summary: check if music player can stop playing with wav format file
Steps: 1. copy sample wav file to system 2. launch music player can play the wav file 3. click "stop" button to stop playing 4. click "start" button to resume playing
Expected Results: wav file can be start/stop without problem

Test Execution Cycle Type:	Weekly
Case Automation	Manual

Type:	
Case State:	Ready
Feature:	multi-media
target:	e-menlow, blacksand, beagleboard, crownbay, sugarbay
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2758: video play (mpeg)	
Summary: make sure video player cannot play mpeg format file	
Steps: 1. copy sample mpeg file to system 2. launch video player and make sure it cannot play the mpeg file	
Expected Results: mpeg file cannot be played	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	multi-media
target:	e-menlow, blacksand, beagleboard, crownbay, sugarbay
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2759: video play (ogg)	
Summary: check if video player can play ogg format file	
Steps: 1. copy sample ogg file to system 2. launch video player can play the ogg file	
Expected Results: ogg file can be played without problem	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	multi-media
target:	e-menlow, blacksand, beagleboard, crownbay, sugarbay
image profile:	sato-sdk
Last Result	**Not Run**

Keywords:	None

Test Case TC-2760: video stop (ogg)	
Summary: check if video player can play ogg format file	
Steps: 1. copy sample ogg file to system 2. launch video player can play the ogg file 3. click "stop" button to stop playing 4. click "start" button to resume playing	
Expected Results: ogg file can be start/stop without problem	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	multi-media
target:	e-menlow, blacksand, beagleboard, crownbay, sugarbay
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2838: audio play (ogg) with HDMI	
Summary: check if music player can play ogg format file when using HDMI	
Steps: 1. copy sample ogg file to system 2. connect system with a monitor with HDMI 3. launch music player and play the ogg file	
Expected Results: ogg file can be played without problem with HDMI	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	multi-media
target:	FRI2
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2839: audio play (wav) with HDMI	
Summary: check if music player can play wav format file with HDMI	
Steps: 1. copy sample wav file to system 2. connect system with a monitor with HDMI 3. launch music player and play the wav file	
Expected Results: wav file can be played without problem, with HDMI	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	multi-media
target:	FRI2
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2840: video play (ogg) with HDMI	
Summary: check if video player can play ogg format file with HDMI	
Steps: 1. copy sample ogg file to system 2. connect system with a monitor with HDMI 3. launch video player and play the ogg file	
Expected Results: ogg file can be played without problem, with HDMI	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	multi-media
target:	FRI2
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

1.9 Test Suite : Compliance

Test Case TC-2761: LTP subset test suite	
Summary: LTP subset test suite	
Steps: For real hardware, run following component, syscalls fs fsx dio io mm ipc sched math nptl pty admin_tools timers commands For QEMU, run following component syscalls mm ipc sched math nptl pty admin_tools commands Run Instructions: LTP download: http://sourceforge.net/projects/ltp/files/LTP%20Source/ltp-20120401/ltp-full-20120401.bz2/download build steps: refer to http://ltp.sourceforge.net Run steps: 1. Build LTP with toolchain or in sdk image 2. For QEMU, create the qemu target with "-m 512", which makes some memory stress cases pass. For some issues, we could only set 128M for qemuarm and 256M for qemumips. 3. Copy LTP folder into target, for example, /opt/ltp. Modify the default scenario file "scenario_groups/default", remove test suites not to be tested 4. Comment runtests/sched: hackbench, which is not suitable to run in emulators 5. Comment creat08 in runtest/syscalls, oom01, oom02, oom03, oom04 in runtest/mm, which consume lots of memory 6. Prepare a tmp folder under your ltp folder, for example, create a tmp folder under your ltp folder, like /opt/ltp/tmp 7. ./runltp -p -l result-M2-20101218.log -C result-M2-20101218.fail -d /opt/ltp/tmp &> result-M2-20101218.fulllog (assume you mount your LTP disk at /opt and create your own tmp dir at /opt/ltp/tmp)	
Expected Results: Check the result on wiki, https://wiki.yoctoproject.org/wiki/LTP_result, there should be no regression failure met.	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Semi-Auto
Case State:	Ready
Feature:	core

target:	qemuarm, qemuppc, qemumips, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, sugarbay
image profile:	sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2762: POSIX subset test suite	
Summary: Run subset test suite of POSIX test suite	
Steps: 1. Get latest LTP sourcecode, download location is http://sourceforge.net/projects/ltp/files/LTP%20Source/ . 2. Go into the folder of LTP, and posix_testsuite is under testcases/open_posix_testsuite/ 3. Run connmand: make generate-makefiles 4. Run connmand: make conformance-all 5. Run connmand: make conformance-test (this step may) 6. Run connmand: make tools-all 7. Run connmand: sh posix.sh &> posix.log, posix.sh as below: ############### #!/bin/sh ./bin/run-posix-option-group-test.sh AIO ./bin/run-posix-option-group-test.sh MEM ./bin/run-posix-option-group-test.sh MSG ./bin/run-posix-option-group-test.sh SEM ./bin/run-posix-option-group-test.sh SIG ./bin/run-posix-option-group-test.sh THR ./bin/run-posix-option-group-test.sh TMR ./bin/run-posix-option-group-test.sh TPS ################ 8. Check the posix.log after testing is finished	
Expected Results: Compare the test result on wiki, https://wiki.yoctoproject.org/wiki/Posix_result, there should be no more regression failures met.	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Semi-Auto
Case State:	Ready
Feature:	core
target:	qemuarm, qemuppc, qemumips, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, sugarbay
image profile:	sato-sdk, lsb-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2763: LSB subset test suite
Summary: Run LSB subset test suite in target
Steps:

1. Get LSB image and start the image(if it is QEMU) with option "-m 512M" 2. Get the LSB test suite or run script creat-lsb-image under poky source directory "scripts/creat-lsb-image" 3. Setup environment for lsb image in target with script LSB_Setup.sh, it could be found under poky source directory "/meta/recipes-extended/lsb/lsbsetup/LSB_Setup.sh" 4. Select LSB test items in LSB web interface and run them
Expected Results: Check the result on wiki, https://wiki.pokylinux.org/wiki/index.php?title=LSB_result&action=edit&redlink=1. No regression failures should be met.

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	core
target:	blacksand, mpc8315e-rdb, sugarbay
image profile:	lsb-sdk
Last Result	**Not Run**
Keywords:	None

1.10 Test Suite : Core Build System

Test Case TC-2772: Incremental RPM image generation
Summary: When modify a package, there is no need to reconstruct the image from scratch, but instead simply use the packaging infrastructure and incrementally update it based on the "package".
Steps: 1. Download poky source and prepare the build environment 2. Add the following line to conf/local.conf: INC_IMAGE_GEN = "1" 3. Run "bitbake core-image-sato" to build a image and check the log.do_rootfs. 4. Remove ${SATO_IMAGE_FEATURES} in meta/recipes-sato/images/core-image-sato.bb. Re-run command "bitbake core-image-sato" and check the log.do_rootfs. 5. Add ${SATO_IMAGE_FEATURES} in meta/recipes-sato/images/core-image-sato.bb. Re-run command "bitbake core-image-sato" and check the log.do_rootfs. 6. Run "bitbake bzip2 -cclean", "rm -f sstate-cache/sstate-bzip2-*", Re-run command "bitbake core-image-sato" and check the log.do_rootfs.
Expected Results: For steps 4,5,6, the log.do_rootfs will show that the rootfs is not reconstruct when some packages changed. Only the modified packages will be added/removed.

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	

image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2777: Archive work directory and export source package	
Summary: Archive work directory and export source package from work directory	
Steps: 1. Download poky source 2. Add following line to meta/classes/package_rpm.bbclass : inherit archive-original-source 3. Prepare the build environment and add the lines to the conf/local.conf: SOURCE_ARCHIVE_PACKAGE_TYPE ?= 'srpm' SOURCE_ARCHIVE_LOG_WITH_SCRIPTS ?= 'logs_with_scripts' 4. Run "bitbake core-image-sato". 5. Change the following lines in conf/local.conf: SOURCE_ARCHIVE_PACKAGE_TYPE ?= 'tar' SOURCE_ARCHIVE_LOG_WITH_SCRIPTS ?= 'logs' Run “bitbake core-image-sato” again.	
Expected Results: The srpm packages or tar packages should be in tmp/deploy/sources after build complete.	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2771: Disk space monitoring	
Summary: Monitor disk availability and warn the user if it is running low	
Steps: 1. Download the source and set build environment. 2. Follow the meta-yocto/conf/local.conf.sample.extended to enable disk monitor. For example, add the following lines to conf/local.conf: BB_DISKMON_DIRS = "STOPTASKS,${TMPDIR},40G,4510K" BB_DISKMON_WARNINTERVAL = "50M,5K" 3. Run "bitbake core-image-sato" to build a image. 4. Change “STOPTASKS” to “ABORT”. Run “bitbake core-image-sato” to build a image. 5. Change “STOPTASKS” to “WARN”. Run “bitbake core-image-sato” to build a image.	
Expected Results: Running "df -h " or "df -i" to check the free space or free inodes of the disk. If “STOPTASKS” is set, when the free disk space or free inodes less than the setting values, the new tasks can't be executed any more, will stop the build when the running tasks have been done. If “ABORTS” is set, when the free disk space or free inodes less than the setting values, the build process would stop immediately. If “WARN” is set, when the free disk space or free inodes less than the setting values, the build	

process would show warnings and repeat the warning when the disk space reduces size BB_DISKMON_WARNINTERVAL	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2778: sanity check for userspace dependency

Summary:

test if sanity check could report warning if there are packages installed under /bin or /sbin, but depends on something under /usr/lib

Steps:

1. prepare yocto build environment
2. modify or add a recipe, which is installed under /bin or /sbin, but depends on something under /usr/lib. For example, we could revert following patch against recipe libusb and build udev

######

```
diff --git a/meta/recipes-support/libusb/libusb-compat_0.1.3.bb b/meta/recipes-
support/libusb/libusb-compat_0.1.3.bb
index ef8552b..e070463 100644
--- a/meta/recipes-support/libusb/libusb-compat_0.1.3.bb
+++ b/meta/recipes-support/libusb/libusb-compat_0.1.3.bb
@@ -15,7 +15,7 @@ DEPENDS = "libusb1"
PROVIDES = "libusb"

PE = "1"
-PR = "r0"
+PR = "r1"

SRC_URI = "${SOURCEFORGE_MIRROR}/libusb/libusb-compat-${PV}.tar.bz2 \
file://0.1.0-beta1-gcc3.4-fix.patch"
@@ -24,3 +24,13 @@ SRC_URI[md5sum] = "570ac2ea085b80d1f74ddc7c6a93c0eb"
SRC_URI[sha256sum] =
"a590a03b6188030ee1ca1a0af55685fcde005ca807b963970f839be776031d94"

inherit autotools pkgconfig binconfig
+
+EXTRA_OECONF = "--libdir=${base_libdir}"
+
+do_install_append() {
+ install -d ${D}${libdir}
+ mv ${D}${base_libdir}/*.la ${D}${libdir}
+ mv ${D}${base_libdir}/pkgconfig ${D}${libdir}
+}
+
+FILES_${PN}-dev += "${base_libdir}/*.so"
```

######

3. check if yocto build will report warning for udev

Expected Results:

test if sanity check could report warning if there are packages installed under /bin or /sbin, but depends on something under /usr/lib	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2779: LICENSE_FLAGS_WHITELIST set for emgd driver build	
Summary: Check if emgd driver could be download automatically with LICENSE_FLAGS_WHITELIST set	
Steps: 1. Prepare yocto build environment 2. Download meta-intel and add crownbay into bblayer.conf 3. Add LICENSE_FLAGS_WHITELIST = "license_emgd-driver-bin_1.10" to local.conf 4. Run "bitbake emgd-driver-bin", and it should run successfully	
Expected Results: emgd driver could be download automatically with LICENSE_FLAGS_WHITELIST set	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2785: lib32 connman-gnome built for qemux86-64 - rpm
Summary: build lib32 connman-gnome and include it in qemux86-64 image
Steps: 1. Prepare poky build environment 2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build and set MACHINE to qemux86-64 as following: ##### MACHINE = "qemux86-64" require conf/multilib.conf MULTILIBS = "multilib:lib32" DEFAULTTUNE_virtclass-multilib-lib32 = "x86" IMAGE_INSTALL_append = "lib32-connman-gnome" ##### 3. with rpm set for package format, build core-image-sato image 4. after build finished, start up the image and check if connman and related packages(like libc) are 32-bit

Expected Results: user could build lib32 connman-gnome and include it in qemux86-64 image	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	core
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2786: lib32 connman-gnome built for qemux86-64 - ipk	
Summary: build lib32 connman-gnome and include it in qemux86-64 image	
Steps: 1. Prepare poky build environment 2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build and set MACHINE to qemux86-64 as following: ##### MACHINE = "qemux86-64" require conf/multilib.conf MULTILIBS = "multilib:lib32" DEFAULTTUNE_virtclass-multilib-lib32 = "x86" IMAGE_INSTALL_append = "lib32-connman-gnome" ##### 3. with ipk set for package format, build core-sato image 4. after build finished, start up the image and check if connman and related packages(like libc) are 32-bit	
Expected Results: user could build lib32 connman-gnome and include it in qemux86-64 image	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	core
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2826: lib32 sato-sdk image build - qemux86-64
Summary: lib32 sato-sdk image should be built out with multilib support
Steps:

1. Prepare poky build environment
2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build and set MACHINE to qemux86-64 as following:
#####
MACHINE = "qemux86-64"
require conf/multilib.conf
MULTILIBS = "multilib:lib32"
DEFAULTTUNE_virtclass-multilib-lib32 = "x86"
#####
3. with rpm set for package format, build lib32-core-sato-sdk image
4. after build finished, start up the image and the kernel should not be able to boot up

Expected Results:

lib32 sato-sdk image should be built out with multilib support

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2828: lib32 lsb-sdk image build - qemux86-64

Summary:

lib32 lsb-sdk image should be built out with multilib support

Steps:

1. Prepare poky build environment
2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build and set MACHINE to qemux86-64 as following:
#####
MACHINE = "qemux86-64"
require conf/multilib.conf
MULTILIBS = "multilib:lib32"
DEFAULTTUNE_virtclass-multilib-lib32 = "x86"
#####
3. with rpm set for package format, build lib32-core-lsb-sdk image
4. after build finished, start up the image and the kernel should not be able to boot up

Expected Results:

lib32 lsb-sdk image should be built out with multilib support

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2787: kernel interactive targets	
Summary: Check if yocto can support kernel interactive target build	
Steps: 1. download yocto source tree 2. prepare yocto build environment 3. Run "bitbake linux-yocto -c menuconfig" 4. Check if a new bash terminal pop up and menuconfig can be triggered	
Expected Results: menuconfig for kernel can be triggered with yocto build command	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2788: KVM enabled with qemu	
Summary: qemu can be started with KVM enabled	
Steps: 1. build a kernel with KVM enabled 2. Start qemu with option "kvm" with runqemu 3. Check if qemu starts up and if kvm_intel is used 4. If kvm_intel is not used when starting qemu, it will shows 0 in "Used by" column when you run "lsmod | grep kvm_intel"	
Expected Results: KVM enabled with qemu	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2797: bitbake-layers show_layers	
Summary:	

show_layers could show current layers
Steps: 1. prepare poky build environment 2. add meta-rt into bblayer.conf 3. run "bitbake-layers show_layers", it should show the layers defined in bblayer.conf
Expected Results: show_layers could show current layers

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2798: bitbake-layers show_overlayed
Summary: overlayed recipes should be shown with bitbake-layers
Steps: 1. prepare poky build environment 2. copy a recipe from meta layer into meta-yocto, for example, /home/jxu49/osel/poky/meta/recipes-graphics/clutter/clutter-1.6_1.6.14.bb 3. run "bitbake-layers show_overlayed", it should report clutter is overlayed by meta-yocto
Expected Results: overlayed recipes should be shown with bitbake-layers

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2799: bitbake-layers show_appends
Summary: bitbake-layers show_appends should list bbappend files and recipe files they apply to
Steps:

1. prepare poky build environment 2. run "bitbake-layers show_appends", it should list bbappend files and recipe files they apply to
Expected Results: bitbake-layers show_appends should list bbappend files and recipe files they apply to

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2800: bitbake-layers flatten
Summary: bitbake-layers flattens layer configuration into a separate output directory
Steps: 1. prepare poky build environment 2. create a folder, for example, test 3. run "bitbake-layers flatten test", all contents of all layers should be moved into the test folder, with any bbappends appended to corresponding recipes 4. check if bbappends take effect, for example, check if test/recipes-bsp/formfactor/formfactor_0.0.bb has the code defined in meta-yocto/recipes-bsp/formfactor/formfactor_0.0.bbappend
Expected Results: bitbake-layers flattens layer configuration into a separate output directory

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2801: x32 image build
Summary: x32 image could be built out successfully
Steps: 1. Prepare yocto build environment 2. add meta-x32 layer, http://git.yoctoproject.org/cgit/cgit.cgi/experimental/meta-x32/

3. Add following lines in your conf/local.conf MACHINE = "qemux86-64" DEFAULTTUNE = "x86-64-x32"
Expected Results: x32 image could be built out successfully

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	core
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2802: x32 image build boot up and check
Summary: x32 image could be built out successfully and binaries/libraries are x32 in it
Steps: 1. Prepare yocto build environment 2. add meta-x32 layer, http://git.yoctoproject.org/cgit/cgit.cgi/experimental/meta-x32/ 3. Add following lines in your conf/local.conf MACHINE = "qemux86-64" DEFAULTTUNE = "x86-64-x32" baselib = "${@d.getVar('BASE_LIB_tune-' + (d.getVar('DEFAULTTUNE', True) or 'INVALID'), True) or 'lib'}" 4. build minimal image with "bitbake core-image-minimal" 5. Run the file command to know what type of elf binary is it. It should be 32bit x86-64 elf binary as seen here: $ file bin/busybox bin/busybox: setuid ELF 32-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.35, not stripped $file usr/lib/libz.so.1.2.5 usr/lib/libz.so.1.2.5: ELF 32-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, not stripped
Expected Results: x32 image could be built out successfully and binaries/libraries are x32 in it

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	core
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2789: non-GPLv3 build check	
Summary: Check if non-GPLv3 build could pass and it does not has any GPLv3 packages installed	

Steps:
 	
1. Set following sentences in local.conf to GPLv3
#####
INCOMPATIBLE_LICENSE = "GPLv3"
#####
2. Build core-image-minimal and core-image-basic
3. Start up target after build is finished
4. Run following script to check if any GPLv3 packages installed, some packages are GPLv3 exception, like libgcc1, libstdc++ and less.
 	
##################
#!/bin/sh
 	
temp=`mktemp`
rpm -qa > $temp
ret=0
 	
for i in `cat $temp`
do
  rpm -qi $i | grep License | grep -i gplv3 > /dev/null 2>&1
  if [$? -eq 0]; then
  license=`rpm -qi $i | grep License | awk -F"License:" '{print $2}'`
  echo "package $i has inconsistent license: $license"
  ret=1
  fi
done
 	
rm -rf $temp
exit $ret
####################	

Expected Results: non-GPLv3 build pass and no GPLv3 packages installed in the image	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2794: sstate work on local host
Summary: Check if sstate could work with local cache
Steps: 1. Follow the wiki steps to setup a sstate cache on local machine, https://wiki.yoctoproject.org/wiki/Enable_sstate_cache

2. Prepare another yocto source directory and set the SSTATE_DIR the cache you setup in step 1) 3. Run poky build, for example, "bitbake core-image-minimal". You should note following things if sstate works: ######## NOTE: Preparing runqueue NOTE: Executing SetScene Tasks NOTE: Running setscene task 118 of 155 (virtual:native:/home/lulianhao/poky-build/edwin/poky/meta/recipes-devtools/pseudo/pseudo_git.bb:do_populate_sysroot_setscene) NOTE: Running setscene task 119 of 155 (/home/lulianhao/poky-build/edwin/poky/meta/recipes-devtools/quilt/quilt-native_0.48.bb:do_populate_sysroot_setscene ########
Expected Results: sstate should work and reduce build time

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2824: ddimage to burn image
Summary: User could use ddimage to burn image into boot media
Steps: 1. prepare yocto build environment 2. get a hddimg for BSP, for example, hddimg for sugarbay 3. use ddimage to burn hddimg into USB stick, "ddimage xxx.hddimg /dev/sdx" 4. check if the USB stick bootable on real board
Expected Results: User could use ddimage to burn image into boot media

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2829: bitbake -b option
Summary:

Check if bitbake -b with a invalid recipe should not print meaningless information
Steps: 1. Prepare yocto build environment 2. Run "bitbake -b test_bitbake", which does not exist in yocto 3. Check if there is any meaningless information printed, for example, python call trace 4. There should be few sentences showing that no recipe matching "test_bitbake"
Expected Results: bitbake -b with a invalid recipe should not print meaningless information

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2830: error message shown at end of bitbake output
Summary: Check if warning and error messages are shown at the end of a bitbake build
Steps: 1. Prepare yocto build environment 2. Run "bitbake man", or anything which could be built out with yocto 3. After build is finished, check the end of the screen output, there should be a summary of how many warnings and errors found with the build
Expected Results: warning and error messages are shown at the end of a bitbake build

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2831: bitbake "NoProvider" message check
Summary: Check if bitbake a invalid recipe shows simple error message
Steps:

1. Prepare yocto build environment 2. Run "bitbake asdf", or anything which does not exist in yocto 3. bitbake should reports a simply summary that there is no provide for the recipe, without many python call trace
Expected Results: there should be no python call trace when bitbake a invalid recipe

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2832: do_patch error report check
Summary: Check if there is no python call trace error when do_patch fail
Steps: 1. Prepare yocto build environment 2. Modify one patch for recipe and make it could not be applied successfully. For example, you could modify the patches for "man" 3. Run "bitbake man -c patch" 4. bitbake should report the error of patching without python call trace
Expected Results: there is no python call trace error when do_patch fail

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2795: btrfs format image build
Summary: btrfs format image could be built out
Steps: 1. set IMAGE_FSTYPES = "btrfs" and KERNEL_FEATURES_append = " cfg/btrfs " in local.conf 2. build a core-image-minimal image, the image should be btrfs format

Expected Results: btrfs format image could be built out	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2796: btrfs format image boot up	
Summary: btrfs format image could be booted up	
Steps: 1. set IMAGE_FSTYPES = "btrfs" and KERNEL_FEATURES_append = " cfg/btrfs " in local.conf 2. build a qemux86 core-image-minimal image and boot up it	
Expected Results: btrfs format image could be booted up	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2814: yocto-bsp list available values for karch	
Summary: User could use yocto-bsp list available values for karch	
Steps: 1.git clone the poky source and setup the build environment 2.Run command "yocto-bsp list karch"	
Expected Results: Several arches supported should be shown,for example, there are x86_64,powerpc,i386,mips,qemu and arm.	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual

Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2815: yocto-bsp list available property values	
Summary: User could use yocto-bsp list all available properties for an arch	
Steps: 1.git clone the poky source and setup the build environment 2.Run command "yocto-bsp list i386 properties"	
Expected Results: A list of properities should be printed	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2818: yocto-kernel add patch
Summary: User could use yocto-kernel to add patches for a BSP kernel recipe
Steps: 1. Follow the case "yocto-bsp create QEMU BSP" to create a QEMU BSP 2. Follow the instruments in https://wiki.yoctoproject.org/wiki/Transcript:_Using_the_Yocto_BSP_tools_to_manage_kernel_patches_and_config_items, to add a patch as below with command "yocto-kernel patch add myqemuarm ~/newpatches/yocto-testmod.patch" 3. use command "yocto-kernel patch list myqemuarm" to show the patch applied to kernel. ######## diff --git a/drivers/misc/Kconfig b/drivers/misc/Kconfig index 6a1a092..b6165b6 100644 --- a/drivers/misc/Kconfig +++ b/drivers/misc/Kconfig @@ -392,6 +392,11 @@ config HMC6352 This driver provides support for the Honeywell HMC6352 compass, providing configuration and heading data via sysfs. +config YOCTO_TESTMOD + tristate "Yocto Test Driver"

```
+       help
+         This driver provides a silly message for testing Yocto.
+
 config EP93XX_PWM
        tristate "EP93xx PWM support"
        depends on ARCH_EP93XX
diff --git a/drivers/misc/Makefile b/drivers/misc/Makefile
index 3e1d801..11384d8 100644
--- a/drivers/misc/Makefile
+++ b/drivers/misc/Makefile
@@ -36,6 +36,7 @@ obj-$(CONFIG_TI_DAC7512)      += ti_dac7512.o
 obj-$(CONFIG_C2PORT)           += c2port/
 obj-$(CONFIG_IWMC3200TOP)      += iwmc3200top/
 obj-$(CONFIG_HMC6352)          += hmc6352.o
+obj-$(CONFIG_YOCTO_TESTMOD)    += yocto-testmod.o
 obj-y                          += eeprom/
 obj-y                          += cb710/
 obj-$(CONFIG_SPEAR13XX_PCIE_GADGET)    += spear13xx_pcie_gadget.o
diff --git a/drivers/misc/yocto-testmod.c b/drivers/misc/yocto-testmod.c
new file mode 100644
index 0000000..81de912
--- /dev/null
+++ b/drivers/misc/yocto-testmod.c
@@ -0,0 +1,36 @@
+/*
+ * Copyright 2012 Intel Corporation
+ * Authored-by:  Tom Zanussi <tom.zanussi@intel.com>
+ *
+ * This program is free software; you can redistribute it and/or modify
+ * it under the terms of the GNU General Public License version 2 as
+ * published by the Free Software Foundation.
+ *
+ * This program is distributed in the hope that it will be useful,
+ * but WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
+ * GNU General Public License for more details.
+ *
+ * You should have received a copy of the GNU General Public License along
+ * with this program; if not, write to the Free Software Foundation, Inc.,
+ * 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
+ */
+
+#include <linux/module.h>
+
+static int __init yocto_testmod_init(void)
+{
+       printk("Kilroy was here! __m_(OuO)_m__");
+}
+
+static void __exit yocto_testmod_exit(void)
+{
+       printk("Kilroy was not here!");
+}
+
+module_init(yocto_testmod_init);
+module_exit(yocto_testmod_exit);
+
+MODULE_AUTHOR("Tom Zanussi <tom.zanussi@intel.com");
+MODULE_DESCRIPTION("Yocto Test Driver");
+MODULE_LICENSE("GPL");
########
```

Expected Results:

User could use yocto-kernel to add patches for a BSP kernel recipe

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual

Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2819: yocto-kernel list config of BSP kernel	
Summary: User could use yocto-kernel to list yocto-kernel config items	
Steps: 1. Follow the case "yocto-bsp create QEMU BSP" to create a QEMU BSP 2. Follow the instruments in https://wiki.yoctoproject.org/wiki/Transcript:_Using_the_Yocto_BSP_tools_to_manage_kernel_patches_and_config_items, to show all the kernel options set for the kernel, with command "yocto-kernel config list myqemuarm"	
Expected Results: A list of kernel options should be shown	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2773: Enumerate possible values for property	
Summary: User could use yocto-bsp to enumerate the possible values for each property	
Steps: 1.git clone the poky source and setup the build environment 2.Use "yocto-bsp list karch property xxx" to show the possible values for the xxx feature. For example, run command "yocto-bsp list i386 properity xserver", which will show all the possible values for xserver	
Expected Results: It will show all the possible values that exist and can be specified for any of the enumerable properties.	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready

Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2821: yocto-kernel remove kernel patch	
Summary: User could use yocto-kernel to remove BSP kernel patch	
Steps: 1. Follow the case "yocto-kernel add patch" apply a patch to your kernel 2. Follow the instruments in https://wiki.yoctoproject.org/wiki/Transcript:_Using_the_Yocto_BSP_tools_to_manage_kernel_patches_and_config_items, to remove the patch with command "yocto-kernel patch rm myqemuarm" 3. check if the patch is removed with command "yocto-kernel patch list myqemuarm"	
Expected Results: User could use yocto-kernel to remove BSP kernel patch	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2823: yocto-bsp create via JSON file
Summary: User could use yocto-bsp to create a new Yocto BSP layer via input a JSON file
Steps: 1. git clone the poky source and setup the build environment 2. create a new qemu Yocot BSP based on i386, with command like :yocto-bsp create myqemux86 qemu -o <DIRNAME> -i <JSON_PROPERTY_ FILE>, <DIRNAME> is the name of BSP dir to create, name of file containing the values for BSP properties as a JSON file. 3. after the new bsp is created, add the new BSP layer to BBLAYERS in bblayers.conf. 4. Edit local.conf set MACHINE to your new machine "myqemux86". 5. Then run "bitbake core-image-sato" and boot the sato image after build is finished. ##################New a file named test.json############### {"kernel_choice":"linux-yocto_3.2","use_default_kernel":"n","keyboard":"y","qemuarch":"i386","smp":"y","touchscreen":"n","need_new_kbranch":"y","new_kbranch":"standard/default/base"} ##################################
Expected Results:

it will create our BSP layer in <DIRNAME> directory, build and boot sato image succeed.

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2764: Init scripts
Summary: Provide an image/recipe skeleton as a canonical example. Check if can be built and run correctly
Steps: Steps: 1. Build image from poky source, check if skeleton script and skeleton-test can be built into the image a. download poky source b. add a new line “ service” ” to the end of “RDEPENDS_task-core-x11-base = "\” section in meta/recipes-sato/tasks/task-core-x11.bb c. $ source oe-init-build-env add line “<POKY_BASE>/meta-skeleton \” to conf/bblayer.conf d. build the image e. boot up the image, check the skeleton and skeleton-test should be in right place /etc/init.d/skeleton /usr/sbin/skeleton-test 2. Verify the basic function of skeleton. Check if skeleton script can start/stop the skeleton-test daemon.
Expected Results: Init scripts can be built and run correctly

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2765: Share gcc work directories
Summary: This feature make gcc use the shared source directory during the different building. Check if this feature can work for gcc 4.5.1 and gcc 4.6.0.

Steps:

1. Download the poky source and set build environment.

2. For gcc 4.5.1, add 2 lines to conf/local.conf :
 GCCVERSION ?= "4.5.1"
 SDKGCCVERSION ?= "4.5.1"
    For gcc 4.6.1, there is no need to add these 2 lines to conf/local.conf

3. Run bitbake command as below:
   bitbake gcc-cross
   bitbake gcc-cross gcc-cross-initial gcc-cross-intermediate -c clean
   bitbake gcc-crosssdk
   bitbake gcc-runtime
   bitbake libgcc
   bitbake gcc-cross-canadian-arm (for arm arch)
   bitbake gcc-cross-canadian-powerpc (for ppc arch)
   bitbake gcc-cross-canadian-mips (for mips arch)

4. Run "bitbake core-image-minimal", "bitbake core-image-sato", "bitbake core-image-sato-sdk" to build images. Verify the basic function of the images.

Expected Results:

After step3, you can check the tmp/work-shared/gcc-4.6.0 or tmp/work-shared/gcc-4.5.1 should in the build directory. Check the time of build process and the disk space usage of tmp/work-shared/gcc-version sub-directory.
The images should be built and can work correctly.

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2766: ccache as native tool

Summary:

ccache - a fast C/C++ compiler cache.

Steps:

1. Make sure the native ccache is not installed on local machine and compile 'less' bbfile without native ccache support.
   bitbake less -c cleansstate
   bitbake less
Check the compile log under .../tmp/work/mips-poky-linux/less-443-r0/temp/log.do_compile

2. Build native tool 'ccache'
   bitbake ccache-native
Check the ccache-native installed location ..tmp/sysroots/x86_64-linux/usr/bin/ccache
Add the following line in conf/local.conf:
   INHERIT += "ccache"

3. Compile less bbfile again with native ccache support
   bitbake less -c cleansstate
   bitbake less -c compile

Check the compile with ccache log under .../tmp/work/mips-poky-linux/less-443-r0/temp/log.do_compile. The native ccache should be used when compiled.	
Expected Results: The ccache-native should be built successfully and be installed to the correct location. The ccache-navive will be used when compile file.	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2767: PAM support	
Summary: Check the Yocto should support PAM (Pluggable Authentication Module)	
Steps: 1. Build a sato-sdk image from poky source with PAM support by following the wiki: https://wiki.yoctoproject.org/wiki/PAM_Integration 2. Refer to https://wiki.yoctoproject.org/wiki/PAM_Integration , check the commands 'dropbear', 'login', 'passwd', 'useradd', 'su' can work correctly with PAM support and verify the function of PAM.	
Expected Results: The commands which have PAM support should run correctly and the function of PAM should work without problems.	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2768: Gtk+ Over DirectFB
Summary: Check if the gtk-directfb image can be built out and gtk-demo can run
Steps: 1. Download poky source and prepare the build environment 2. Set MACHINE to qemuarm in conf/local.conf 3. Remove "x11" from DISTRO_FEATURES in meta/conf/distro/include/default-distrovars.inc, use "gtk-directfb" instead of it: DISTRO_FEATURES ?= "alsa argp bluetooth ext2 irda largefile pcmcia usbgadget usbhost wifi xattr nfs zeroconf pci 3g gtk-directfb ${DISTRO_FEATURES_LIBC}" 4. Run "bitbake core-image-gtk-directfb" to build a gtk-directfb image

5. Boot up the gtk-directfb image and run "gtk-demo" command.	
Expected Results: The gtk-directfb image can be built out and the "gtk-demo" command can run without problems.	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2769: bitbake-runtask	
Summary: Check if bitbake-runtask command could work.	
Steps: 1. Download poky source and prepare the build environment . 2. Run “bitbake-runtask” command to build some packages. For example, run the following commands: "bitbake-runtask man_1.6f do_fetch" "bitbake-runtask man_1.6f do_unpack" "bitbake-runtask man_1.6f do_patch" "bitbake-runtask man_1.6f do_configure" "bitbake-runtask man_1.6f do_compile" "bitbake-runtask man_1.6f do_install" "bitbake-runtask man_1.6f do_populate_lic" "bitbake-runtask man_1.6f do_populate_sysroot" 3. Check the return value of each command by using “echo $?” and check the log file in work directory.	
Expected Results: The return value of each command should be “0” and no error message in log file.	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2770: autoconf-nativesdk and automake-nativesdk support
Summary: Check if toolchain support autoconf-nativesdk and automake-nativesdk.
Steps:

1. Install toolchain tarball and setup cross compile environment.
2. Check if there are “autoconf” and “automake” commands in toolchain tarball. Check if there is a option “--with-libtool-sysroot” in ${CONFIGURE_FLAGS}.
3. Download iptables project. There is a macro “AM_PROG_LIBTOOL” in configure.ac. With the cross compile environment, run "autoreconf", "./configure ${CONFIGURE_FLAGS}", "make", "make install DESTDIR=/opt/tmp"

Expected Results:

The “autoconf” and “automake” commands should be contained in meta-toolchain. When running “./configure ${CONFIGURE_FLAGS}”, there is no warning message like:
“WARNING: unrecognized options: --with-libtool-sysroot”

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2774: Clean obsolete sstate cache files

Summary:

Check if the script could clean the obsolete sstate cache files

Steps:

1. Download poky source and prepare the build environment.
2. Follow the wiki steps to setup a sstate cache on local machine, https://wiki.yoctoproject.org/wiki/Enable_sstate_cache
3. Run "bitbake core-image-minimal" to build a image.
4. Set MACHINE to another architecture. Run "bitbake core-image-minimal" to build a image.
5. Run script ../scripts/sstate-cache-management.sh --cache-dir=sstate-cache --remove-duplicated and check the ouput.

Expected Results:

The obsolete sstate cache files should be removed.

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2775: Enable cleanup of WORKDIR

Summary:

A script that could go through and prune out old versions of recipes in WORKDIR

Steps:

1. Download poky source and prepare the build environment
2. Run “bitbake gzip” . The gzip 1.4 would be built.
3. Run “bitbake -b ../meta/recipes-extended/gzip/gzip_1.3.12.bb”. The gizp 1.3.12 would be built.
4. Run “../scripts/cleanup-workdir”

Expected Results:

The old version of gizp would be cleanup. Only the latest version would be left in WORKDIR.

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2776: Allow logrotate to use a different file system

Summary:

Allow logrotate to use a different file system from the original logs

Steps:

1. Download poky source and prepare the build environment
2. Add the lines to the conf/local.conf:
DISTRO_EXTRA_RDEPENDS += "logrotate"
3. Run "bitbake core-image-sato".
4. Boot up the image and add the following lines to /etc/logroate.conf:
/var/log/wtmp {
monthly
create 0664 root utmp
minsize 1M
olddir /home/root/logrotate_dir
rotate 1
}
Make sure the directory "/home/root" is on a different filesystem.
5. Run "logrotate -f /etc/logrotate.conf".

Expected Results:

The logs would be compressed in direcotry “/home/root” on a different filesystem

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2780: lib64 sato image build - qemux86-64/ipk

Summary: lib64 sato image should be built out with multilib support
Steps: 1. Prepare poky build environment 2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build and set MACHINE to qemux86-64 as following: ##### MACHINE = "qemux86-64" require conf/multilib.conf MULTILIBS = "multilib:lib64" DEFAULTTUNE_virtclass-multilib-lib64 = "x86-64" ##### 3. with ipk set for package format, build lib64-core-image-sato image 4. after build finished, start up the image and check if all app are 64-bit, kernel with 64-bit
Expected Results: lib64 sato image should be built out with multilib support

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2781: lib64 sato image build - qemux86-64
Summary: lib64 sato image should be built out with multilib support
Steps: 1. Prepare poky build environment 2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build and set MACHINE to qemux86-64 as following: ##### MACHINE = "qemux86-64" require conf/multilib.conf MULTILIBS = "multilib:lib64" DEFAULTTUNE_virtclass-multilib-lib64 = "x86-64" ##### 3. with rpm set for package format, build lib64-core-image-sato image 4. after build finished, start up the image and check if all app are 64-bit, kernel with 64-bit
Expected Results: lib64 sato image should be built out with multilib support

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system

image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2782: lib64 sato image build - qemux86	
Summary: lib64 sato image should be built out with multilib support	
Steps: 1. Prepare poky build environment 2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build and set MACHINE to qemux86 as following: ##### MACHINE = "qemux86" require conf/multilib.conf MULTILIBS = "multilib:lib64" DEFAULTTUNE_virtclass-multilib-lib64 = "x86-64" ##### 3. with rpm set for package format, build lib64-core-image-sato image 4. after build finished, start up the image and the kernel should not be able to boot	
Expected Results: lib64 sato image should be built out with multilib support	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2783: lib32 sato image build - qemux86-64
Summary: lib32 sato image should be built out with multilib support
Steps: 1. Prepare poky build environment 2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build and set MACHINE to qemux86-64 as following: ##### MACHINE = "qemux86-64" require conf/multilib.conf MULTILIBS = "multilib:lib32" DEFAULTTUNE_virtclass-multilib-lib32 = "x86" ##### 3. with rpm set for package format, build lib32-core-image-sato image 4. after build finished, start up the image and check if all app are 32-bit, kernel with 64-bit
Expected Results: lib32 sato image should be built out with multilib support

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2784: lib32 sato image build - qemux86	
Summary: lib32 sato image should be built out with multilib support	
Steps: 1. Prepare poky build environment 2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build and set MACHINE to qemux86 as following: ##### MACHINE = "qemux86" require conf/multilib.conf MULTILIBS = "multilib:lib32" DEFAULTTUNE_virtclass-multilib-lib32 = "x86" ##### 3. with rpm set for package format, build lib32-core-image-sato image 4. after build finished, start up the image and check if all app are 32-bit, kernel with 32-bit	
Expected Results: lib32 sato image should be built out with multilib support	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2793: yocto build in KVM
Summary: Build yocto in KVM should work
Steps: 1. Setup a VM environment with KVM enabled, for example, RHEL6 2. Prepare a VM for yocto build testing, for example, OpenSuse 11.3 3. By following the yocto handbook, download latest yocto source into the VM 4. Build core-image-minimal in the VM
Expected Results:

Yocto build in VM should work same as in real host

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2803: minimal build with self-hosted-image with vmdk	
Summary: check if self-hosted-image could pass minimal build with vmdk	
Steps: 1. Get poky source code and prepare the build environment 2. Set MACHINE to qemux86-64 and run "bitbake self-hosted-image" 3. After build is finished, start VMWare Player and start the vmdk image with it 4. Build a minimal image in the self-hosted image	
Expected Results: self-hosted-image could pass minimal build with vmdk	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2804: hob launch against self-hosted-image	
Summary: check if self-hosted-image could launch hob	
Steps: 1. Get poky source code and prepare the build environment 2. Set MACHINE to qemux86-64 and run "bitbake self-hosted-image" 3. After build is finished, start VMWare Player and setup poky build environment with self-hosted-image 4. Launch hob in self-hosted-image	
Expected Results: hob could be launched against self-hosted-image	
Test Execution	Fullpass

Cycle Type:	
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2805: bitbake fetch against self-hosted-image	
Summary:	
check if bitbake fetch could work against self-hosted-image	
Steps:	
1. Get poky source code and prepare the build environment 2. Set MACHINE to qemux86-64 and run "bitbake self-hosted-image" 3. After build is finished, start VMWare Player and setup poky build environment in the self-hosted-image, setup the correct proxy for git,wget 4. run "bitbake man -c fetch", "bitbake oprofileui -c fetch" and check if these packages could be downloaded	
Expected Results:	
bitbake fetch could work against self-hosted-image	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2806: PR service enable with remote server
Summary:
enable PR service with remote server/local client mode
Steps:

1. prepare 2 poky build environments
2. in one of the poky source, run "bitbake-prserv --start"
3. in the second poky source, set PRSERV_HOST to 127.0.0.1 and PRSERV_PORT to 8585
4. run "bitbake man" and then add following lines into man_${PV}.bb
#######
do_package_append() {
bb.build.exec_func('do_test_prserv', d)
}
 	
do_test_prserv() {
echo "Test if PR service could work"

} ####### 5. re-run "bitbake man", task do_packge for recipe man should be re-run 6. check the man package built out under deploy folder, the RP number should bump up automatically
Expected Results: RP number should bump up with remote server/local client mode

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2807: PR service enable with local server
Summary: PR service should work with local server/local client

Steps:
 	
1. prepare 1 poky build environments
2. poky source, set PRSERV_HOST to localhost and PRSERV_PORT to 0 in local.conf
4. run "bitbake man" and then add following lines into man_${PV}.bb
#######
do_package_append() {
bb.build.exec_func('do_test_prserv', d)
}
 	
do_test_prserv() {
echo "Test if PR service could work"
}
#######
5. re-run "bitbake man", task do_packge for recipe man should be re-run
6. check the man package built out under deploy folder, the RP number should bump up automatically

Expected Results: PR service should work with local server/local client

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2808: basichash enabled with PR service
Summary:

make sure basichash works with PR service

Steps:

1. prepare 1 poky build environments
2. poky source, set PRSERV_HOST to localhost and PRSERV_PORT to 0 in local.conf
4. run "bitbake man"
5. check the stamps folder, note down the file name of the do_package file for man
6. add following lines into man_${PV}.bb

```
#######
do_package_append() {
       bb.build.exec_func('do_test_prserv', d)
}

do_test_prserv() {
 echo "Test if PR service could work"
}
#######
```

7. re-run "bitbake man", task do_packge for recipe man should be re-run
8. check the stamps folder, the do_package file for man should be regentered with hash value changed

Expected Results:

make sure basichas works with PR service

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2809: AUTOPR export/lockdown

Summary:

check if AUTOPR could be export/lockdown for package build

Steps:

1. prepare 2 poky build environments
2. in one of the poky source, set PRSERV_HOST to localhost and PRSERV_PORT to 0 in local.conf
4. run "bitbake man" and then add following lines into man_${PV}.bb

```
#######
do_package_append() {
       bb.build.exec_func('do_test_prserv', d)
}

do_test_prserv() {
 echo "Test if PR service could work"
}
#######
```

7. re-run "bitbake man", and check the deploy folder if the packages for man are re-generated with PR number bump up
8. run "bitbake-prserv-tool export export.inc"
9. in the second poky source, set PRSERV_HOST to localhost and PRSERV_PORT to 0 in local.conf
10. run "bitbake -R export.inc man"
11. check the deploy folder if the packages for man are generated with same PR number in first

poky build folder	
Expected Results: check if AUTOPR could be export/lockdown for package build	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2810: AUTOPR export/import	
Summary: check if AUTOPR could be export/import for package build	

Steps:
 	
1. prepare 2 poky build environments
2. in one of the poky source, set PRSERV_HOST to localhost and PRSERV_PORT to 0 in local.conf
4. run "bitbake man" and then add following lines into man_${PV}.bb
#######
do_package_append() {
bb.build.exec_func('do_test_prserv', d)
}
 	
do_test_prserv() {
echo "Test if PR service could work"
}
#######
7. re-run "bitbake man", and check the deploy folder if the packages for man are re-generated with PR number bump up
8. run "bitbake-prserv-tool export export.inc"
9. in the second poky source, set PRSERV_HOST to localhost and PRSERV_PORT to 0 in local.conf
10. run "bitbake-prserv-tool import export.inc" and run "bitbake man"
11. check the deploy folder if the packages for man are generated with N+1 PR number compared with the first poky source	

Expected Results: check if AUTOPR could be export/import for package build	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2811: buildhistory enable for yocto build

Summary: check if buildhistory could work during yocto build	
Steps: 1. build of an image (e.g. core-image-minimal) runs through successfully with it enabled (i.e. with INHERIT += "buildhistory" and BUILDHISTORY_COMMIT = "1" in local.conf). 2. Once a build with package history enabled has finished, verify that the output can be found in TMPDIR/buildhistory.	
Expected Results: package information should be under TMPDIR/buildhistory	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2812: buildhistory error if do_package backwards	
Summary: check if buildhistory reports error if PR of some recipes go backwards	
Steps: 1. build some recipes and get the buildhistory log(for example, recipe "man") 2. change the PR of man backwards, for example from "r1" to "r0" 3. re-build the recipe	
Expected Results: pkghistory reports error if PR of some recipes go backwards	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2813: buildhistory-diff for build analysis
Summary: use buildhistory-diff to analyse changes for 2 builds
Steps:

1. build some recipes and get the buildhistory log(for example, recipe "man") with INHERIT += "buildhistory" and BUILDHISTORY_COMMIT = "1" in local.conf 2. change the PR of man backwards, for example from "r1" to "r0" 3. re-build the recipe and run buildhistory-diff to check if there is any change
Expected Results: buildhistory-diff could show changes for 2 builds

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2816: yocto-bsp create QEMU BSP
Summary: User could use yocto-bsp to create a new Yocto BSP layer
Steps: 1. git clone the poky source and setup the build environment 2. follow the instruments on https://wiki.yoctoproject.org/wiki/Transcript:_Using_the_Yocto_BSP_tools_to_create_a_qemu_BSP, create a new qemu Yocot BSP based on i386, with command like :yocto-bsp create myqemux86 qemu, yocto-bsp will ask user to set value for each of the unspecified property, select default option for all of then 3. after the new bsp is created, add the new BSP layer to BBLAYERS in bblayers.conf. 4. Edit local.conf set MACHINE to your new machine "myqemux86". 5. Then run "bitbake core-image-sato" and boot the sato image after build is finished
Expected Results: With the prompt message, it will create our BSP layer in meta-myqemux86 in the current directory, build and boot sato image succeed.

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2817: ycoto-bsp create a meta-intel BSP
Summary: User could use yocto-bsp to create a meta-intel BSP

Steps: 1. git clone the poky source and setup the build environment 2. follow the instruments on https://wiki.yoctoproject.org/wiki/Transcript:_Using_the_Yocto_BSP_tools_to_create_a_meta-intel_BSP, create a new meta-intel Yocot BSP based on x86_64, with command like :yocto-bsp create myintelbsp x86_64, yocto-bsp will ask user to set value for each of the unspecified property, select default option for all of then 3. after the new bsp is created, add the new BSP layer to BBLAYERS in bblayers.conf. 4. Edit local.conf set MACHINE to your new machine "myintelbsp". 5. Then run "bitbake core-image-sato" and burn/boot it after build is finished	
Expected Results: With the prompt message, it will create our BSP layer in meta-myqemux86 in the current directory, build and boot sato image succeed.	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2820: yocto-kernel set kernel config	
Summary: User could use yocto-kernel to set kernel config	
Steps: 1. Follow the case "yocto-kernel add patch" apply a patch to your kernel 2. Follow the instruments in https://wiki.yoctoproject.org/wiki/Transcript:_Using_the_Yocto_BSP_tools_to_manage_kernel_patches_and_config_items, to enable some kernel options for your kernel. 3. For example, run "yocto-kernel config add myqemuarm CONFIG_MISC_DEVICES=y" and "yocto-kernel config add myqemuarm CONFIG_YOCTO_TESTMOD=y" will make CONFIG_MISC_DEVICES and CONFIG_YOCTO_TESTMOD set for kernel 4. Rebuild the kernel and boot from the kernel, check if there is a line with 'Kilroy was here! __m_(OuO)_m__' in command dmesg	
Expected Results: User could use yocto-kernel to set kernel config	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2822: yocto-kernel remove kernel option	
Summary: User could use yocto-kernel to remove kernel option for BSP kernel	
Steps: 1. Follow the case "yocto-kernel set kernel config" to enable some options for BSP kernel 2. Follow the instruments in https://wiki.yoctoproject.org/wiki/Transcript:_Using_the_Yocto_BSP_tools_to_manage_kernel_patches_and_config_items, to disable these options for BSP kernel. For example, CONFIG_MISC_DEVICES and CONFIG_YOCTO_TESTMOD. 3. Rebuild the kernel and boot from it. Check "dmesg" if there is these options are removed or not.	
Expected Results: User could use yocto-kernel to remove kernel option for BSP kernel	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2825: lib64 sato-sdk image build - qemux86	
Summary: lib64 sato-sdk image should be built out with multilib support	
Steps: 1. Prepare poky build environment 2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build and set MACHINE to qemux86 as following: ##### MACHINE = "qemux86" require conf/multilib.conf MULTILIBS = "multilib:lib64" DEFAULTTUNE_virtclass-multilib-lib64 = "x86-64" ##### 3. with rpm set for package format, build lib64-core-sato-sdk image 4. after build finished, start up the image and check if all app are 64-bit, kernel with 32-bit	
Expected Results: lib64 sato-sdk image should be built out with multilib support	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2827: lib64 lsb-sdk image build - qemux86	
Summary: lib64 lsb-sdk image should be built out with multilib support	
Steps: 1. Prepare poky build environment 2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build and set MACHINE to qemux86 as following: ##### MACHINE = "qemux86" require conf/multilib.conf MULTILIBS = "multilib:lib64" DEFAULTTUNE_virtclass-multilib-lib64 = "x86-64" ##### 3. with rpm set for package format, build lib64-core-lsb-sdk image 4. after build finished, start up the image and check if all app are 64-bit, kernel with 32-bit	
Expected Results: lib64 lsb-sdk image should be built out with multilib support	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
Last Result	**Not Run**
Keywords:	None

1.11 Test Suite : BSP specific

Test Case TC-2833: EFI boot	
Summary: check if EFI booting is supported by Intel BSPs	
Steps: 1. Download EFI BSP images from autobuilder or build them on local machine 2. Burn the images into harddisk 3. boot from harddisk and choose EFI shell to boot from EFI 4. check system could boot up with EFI	
Expected Results: check if EFI booting is supported by Intel BSPs	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready

Feature:	bsp
target:	e-menlow, blacksand, crownbay, sugarbay, jasperforest, FRI2
image profile:	
Last Result	**Not Run**
Keywords:	None

Test Case TC-2834: RTC	
Summary: Check if RTC(Real Time Clock) can work correctly	
Steps: 1. Read time from RTC registers. root@localhost:/root> hwclock -r Sun Mar 22 04:05:47 1970 -0.001948 seconds 2. Set system current time root@localhost:/root> date 062309452008 3. Synchronize the system current time to RTC registers root@localhost:/root> hwclock -w 4. Read time from RTC registers root@localhost:/root> hwclock -r 5. Reboot target and read time from RTC again.	
Expected Results: Can read and set the time successful	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	bsp
target:	beagleboard, mpc8315e-rdb
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2835: Watchdog
Summary: Check if watchdog can reset the target system
Steps:

1. Check if watchdog device exist in /dev/ directory

2. Run command “echo 1 > /dev/watchdog” and wait for 60s. Then the target will reboot.

Expected Results:

The watchdog device exist in /dev/ directory and can reboot the target.

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	bsp
target:	beagleboard, routerstationpro
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2836: SATA

Summary:

Test general use of SATA device on target, like mount, umount, read and write.

Steps:

1. Run “fdisk” command to create partition on SATA disk.

2. Mount/Umount

mke2fs /dev/sda1

mount -t ext2 /dev/sda1 /mnt/disk

umount /mnt/disk

3. Read/Write (filesystem)

touch /mnt/disk/test.txt

echo “abcd” > /mnt/disk/test.txt

cat /mnt/disk/test.txt

4. Read/Write (raw)

dd if=/dev/sda1 of=/tmp/test bs=1k count=1k

This command will read 1MB from /dev/sda1 to /tmp/test

Expected Results:

The SATA device can mount, umount, read and write

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	bsp
target:	mpc8315e-rdb
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Test Case TC-2837: I2C/EEPROM	
Summary: Check if target can support EEPROM	
Steps: 1. Check eeprom device exist in /sys/bus/i2c/devices/ 2. Run "hexdump eeprom" command root@mpc8315e-rdb:/sys/bus/i2c/devices/1-0051> hexdump eeprom 0000000 9210 0b02 0211 0009 0b52 0108 0c00 3c00 0000010 6978 6930 6911 208c 7003 3c3c 00f0 8381 1. Check eeprom device exist in /sys/bus/i2c/devices/ 2. Run "hexdump eeprom" command root@mpc8315e-rdb:/sys/bus/i2c/devices/1-0051> hexdump eeprom 0000000 9210 0b02 0211 0009 0b52 0108 0c00 3c00 0000010 6978 6930 6911 208c 7003 3c3c 00f0 8381	
Expected Results: Hexdump can read data from eeprom	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	bsp
target:	mpc8315e-rdb
image profile:	sato-sdk
Last Result	**Not Run**
Keywords:	None

Reports and Metrics