



Yocto 1.2 M2 Fullpass Test Test Report

Project: yocto

Author: admin

Printed by TestLink on 11/01/2012

2009 © Testlink Community

1 Test Suite : Yocto 1.2 M2 Fullpass Test

1.1 Test Suite : hob

Test Case TC-1719: hob launch without error	
<u>Summary:</u>	
hob could be launched without error	
<u>Steps:</u>	
1. Prepare poky build environment 2. launch hob with command "hob" 3. Check if hob is launched correctly and no error message in console	
<u>Expected Results:</u>	
hob launched correctly and no error message	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1720: add layer for new target build	
<u>Summary:</u>	
user could add layer for new target build	
<u>Steps:</u>	
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	
<u>Expected Results:</u>	
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:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1721: base image selectionSummary:

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:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1722: recipe list re-load for "base image" changeSummary:

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 if changing image type for "base image"

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

Test Case TC-1723: recipe list re-load for "Machine" changeSummary:

recipe list for should be re-loaded and correct when "Machine" changing	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	
recipe list should be re-loaded and correct when "Machine" changing	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1724: No native recipe shown in recipe list	
<u>Summary:</u>	
There should be no native recipe shown in recipe list	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	
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:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1725: search recipe name in recipe list	
<u>Summary:</u>	
User could search recipe name from "Search"	
<u>Steps:</u>	

<ol style="list-style-type: none"> 1. launch hob 2. select one "Machine", for example, qemumips 3. click the icon for "View Recipes", then search recipe via "Search" 4. the searched recipe should be shown up 	
<u>Expected Results:</u>	
User could search recipe name from "Search"	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1726: task list re-load when base image change	
<u>Summary:</u>	
task list for "recipe collections" should be re-loaded when base image changing	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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"-">"Recipe Collections", 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 	
<u>Expected Results:</u>	
task list for "recipe collections" 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:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1727: user could customize threads of bitbake and make	
<u>Summary:</u>	
user could customize threads of bitbake and make in hob	
<u>Steps:</u>	

1. launch hob
2. select one "Machine", for example, qemu86
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:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1731: 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, qemu86
3. choose one "Base image", for example, core-image-minimal
4. click "build image" 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:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1728: ipk package build for image/package build

Summary:

build image with ipk package format

Steps:

<ol style="list-style-type: none"> 1. launch hob 2. select one "Machine", for example, qemu86 3. in "Advanced Settings", select ipk for "packaging format" 4. click "Save" and select one image, for example, "core-image-basic" 5. click "build image" button and it should build recipes with ipk format 	
<u>Expected Results:</u>	
build image with ipk package format	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1729: deb package build for image/package build	
<u>Summary:</u>	
build image with deb package format	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. launch hob 2. select one "Machine", for example, qemu86 3. in "Advanced Settings", select deb for "packaging format" 4. click "Save" and select one image, for example, "core-image-basic" 5. click "build image" button and it should build recipes with deb format 	
<u>Expected Results:</u>	
build image with deb package format	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1730: rpm package build for image/package build	
<u>Summary:</u>	
build image with rpm package format	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. launch hob 	

2. select one "Machine", for example, qemux86 3. in "Advanced Settings", select rpm for "packaging format" 4. click "Save" and select one image, for example, "core-image-basic" 5. click "build image" button and it should build recipes with rpm format	
<u>Expected Results:</u>	
build image with rpm package format	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1909: multiple package format set for build	
<u>Summary:</u>	
build image with multiple package format set	
<u>Steps:</u>	
1. launch hob 2. select one "Machine", for example, qemux86 3. in "Advanced Settings", select all 3 options for "packaging format", rpm, ipk, deb 4. click "Save" and select one image, for example, "core-image-basic" 5. click "build image" button and it should build recipes with rpm, ipk, deb format	
<u>Expected Results:</u>	
build image with multiple package format set	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
target:	
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1869: recipe list reset	
<u>Summary:</u>	
reset button should clear recipe list	
<u>Steps:</u>	
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. click reset button and all recipes should be deselected	
<u>Expected Results:</u>	
reset button should clear recipe list	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1900: stop build during image/package building	
<u>Summary:</u>	
"stop build" button should be able to stop/force stop building	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 "x" button, then click "stop" or "force stop" to stop the build 	
<u>Expected Results:</u>	
"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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1901: template file save/load	
<u>Summary:</u>	
user could save customized template file and load it in hob	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. launch hob 2. select one "Machine", for example, qemuarm 3. choose one "Base image", for example, core-image-basic 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. de-select some selected recipe, for example, zypper 6. click "build image" button and it should show a build progress bar 7. after build finished successfully, click the "Save Template Files" button to save the build information into a template file 8. re-launch hob and click "Load Template Files" and choose the template file saved as above 9. The user customized recipe list should be shown in "View Recipes" 	

<u>Expected Results:</u>	
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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1902: another build after stop build	
<u>Summary:</u>	
user could start another build after stop a build	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 "x" 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 	
<u>Expected Results:</u>	
user could start another build after stop a build	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1903: build a image without error(base image)	
<u>Summary:</u>	
user could use hob to build a image without error	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. launch hob 2. select one "Machine", for example, qemuarm 3. choose one "Base image", for example, core-image-minimal 4. click "build image" button and wait for a successful build finished 	
<u>Expected Results:</u>	
user could use hob to build a image without error	
Test Execution Cycle Type:	Fullpass
Case Automation	Manual

Type:	
Case State:	Ready
Feature:	hob
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1904: build a image without error (added recipe)	
<u>Summary:</u>	
user could use hob to build a image without error	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 image" button and wait for a successful build finished 6. after build finished, check if the added recipe built into image 	
<u>Expected Results:</u>	
user could use hob to build a image without error	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1905: build a image without error (remove recipe)	
<u>Summary:</u>	
user could use hob to build a image without error	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. launch hob 2. select one "Machine", for example, qemuarm 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, deselect some selected recipe, for example, zypper 5. click "build image" button and wait for a successful build finished 6. after build finished, check if the removed recipe not built into image 	
<u>Expected Results:</u>	
user could use hob to build a image without error	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob

<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1906: toolchain built correct with user customization	
<u>Summary:</u>	
toolchain generated correct with user selection	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. launch hob 2. select one "Machine", for example, qemumips 3. choose one "Base image", for example, core-image-sato 4. click icon for "Advanced Settings", and select "Build Toolchain", for toolchain host, you could pick up one, for example, x86_64 5. click "build image" button and wait for a successful build finished 6. after build finished, check if toolchain is built out with the correct host/target arch 	
<u>Expected Results:</u>	
toolchain generated correct with user selection	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1907: non-GPLv3 build	
<u>Summary:</u>	
non-GPLv3 build should be supported for hob	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 "Advanced Settings", and select "Exclue GPLv3 packages" 5. click "build image" button and wait for a successful build finished 	
<u>Expected Results:</u>	
non-GPLv3 build should be supported for hob	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1908: distribution selection for build	
<u>Summary:</u>	
user could select different distribution for "distribution"	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. launch hob 2. select one "Machine", for example, qemumips 3. choose one "Base image", for example, core-image-minimal 4. click icon for "Advanced Settings", and select different distribution for "Select Distro", for example, poky-lsb 5. click "build image" button and wait for a successful build finished 	
<u>Expected Results:</u>	
user could select different distribution for "distribution"	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	hob
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

1.2 Test Suite : System & Core OS

Test Case TC-1732: zypper command installed and workable	
<u>Summary:</u>	
check if zypper is installed and can work	
<u>Steps:</u>	
1. Run command "zypper", and check the output	
<u>Expected Results:</u>	
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, jasperformest
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1733: zypper help search	
<u>Summary:</u>	
check help option with zypper command	
<u>Steps:</u>	
1. Run "zypper help search" and check the output	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1734: zypper search package	
<u>Summary:</u>	
search package with zypper	
<u>Steps:</u>	
1. Run "zypper search package_name" and check the output, for example "zypper search avahi"	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1735: zypper remove package	
<u>Summary:</u>	
remove package with zypper	
<u>Steps:</u>	
1. Run "zypper rm package_name" and check the output, for example "zypper rm avahi"	

<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1736: zypper install package	
<u>Summary:</u>	
install package with zypper	
<u>Steps:</u>	
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	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1737: zypper install dependency package	
<u>Summary:</u>	
install dependency package with zypper	
<u>Steps:</u>	

<p>1. Set up a yum based repository on local server</p> <p>2. Build out a package, which does not need any run-time dependency package, with local poky tree. For example, package "mc"</p> <p>3. In target system, run "zypper addrepo http://ip_address_of_repository zypper_test_repo"</p> <p>4. Run "zypper refresh" to refresh the zypper repository cache</p> <p>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.</p>	
<u>Expected Results:</u>	
The command should install package "mc" and dependency 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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1738: zypper install .all packages	
<u>Summary:</u>	
install packages from all folder with zypper	
<u>Steps:</u>	
<p>1. Set up a yum based repository on local server</p> <p>2. Build out a package, which belongs to all folder, for example, xcursor-transparent-theme-dbg-0.1.1-r3.all.rpm.</p> <p>3. In target system, run "zypper addrepo http://ip_address_of_repository zypper_test_repo"</p> <p>4. Run "zypper refresh" to refresh the zypper repository cache</p> <p>5. Run "zypper install xcursor-transparent-theme-dbg" and check the output</p>	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1739: rpm query package	
<u>Summary:</u>	
make sure rootfs image is built with rpm packages	
<u>Steps:</u>	
1. launch terminal	
2. run command "rpm -qa", which lists all existing packages in system	
<u>Expected Results:</u>	
"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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1740: rpm install package	
<u>Summary:</u>	
rpm format package can be installed	
<u>Steps:</u>	
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	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1741: rpm install dependency package

<u>Summary:</u>	
rpm command should report dependency when installing package	
<u>Steps:</u>	
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	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1742: rpm remove package	
<u>Summary:</u>	
rpm command can remove package in system	
<u>Steps:</u>	
1. Launch terminal and run command "rpm -e package_name" to remove some package, for example, avahi	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1743: check rpm install/removal log file size	
<u>Summary:</u>	

The case is to track log file size after rpm install/removal	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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. 	
<u>Expected Results:</u>	
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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1744: boot and install from USB	
<u>Summary:</u>	
boot and install image from usb stick	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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. 	
<u>Expected Results:</u>	
<ol style="list-style-type: none"> 1. User can choose install system from usb stick onto harddisk from boot menu or command line option 2. Installed system can boot up 	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	installation&boot
target:	e-menlow, blacksand, crownbay, sugarbay, jasperforest
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1745: live boot from USB	
<u>Summary:</u>	

live boot from USB	
<u>Steps:</u>	
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	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1746: boot from runlevel 3	
<u>Summary:</u>	
Verify that system can boot from runlevel 3	
<u>Steps:</u>	
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	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1747: 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
image profile:	sato, sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1748: 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); // or double(l)
```

<pre> } 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; } ##### </pre>	
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
image profile:	sato-sdk, lsb-sdk
Last Result	Not Run
Keywords:	None

Test Case TC-1749: gcc compile in sdk image	
<u>Summary:</u>	
check if gcc can compile program in sdk image	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 	
<pre> 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; </pre>	

<pre>double f; f = convert(l); printf("convert: %lld => %f\n", l, f); f = 1234.67; printf("floor(%f) = %f\n", f, floor(f)); return 0; } #####</pre>	
<u>Expected Results:</u>	
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
image profile:	sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1750: run command make in sdk image	
<u>Summary:</u>	
check if command make can work in sdk image	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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" 	
<pre>test: test.o gcc -o test test.o -lm test.o: test.c gcc -c test.c</pre>	
<u>Expected Results:</u>	
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
image profile:	sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run

<u>Keywords:</u>	None
------------------	------

Test Case TC-1751: cvs project compile in sdk image	
<u>Summary:</u>	
cvs project could be compiled in sdk image	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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" 	
<u>Expected Results:</u>	
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
image profile:	sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1752: iptables project compile in sdk image	
<u>Summary:</u>	
iptables project could be compiled in sdk image	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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" 	
<u>Expected Results:</u>	
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
image profile:	sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1753: sudoku-savant project compile in sdk image	
<u>Summary:</u>	
sudoku-savant could be compiled in sdk image	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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" 	
<u>Expected Results:</u>	
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
image profile:	sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1754: perl program work in image	
<u>Summary:</u>	
A perl program could be executed and output correctly in image	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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" 	
<pre>##### \$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"); #####</pre>	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1755: shutdown system	
<u>Summary:</u>	
verify that system can be shutdown by command	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. boot system 2. launch terminal and run "shutdown -h now" or "poweroff" 	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1756: reboot system	
<u>Summary:</u>	
verify that system can boot by command	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. boot system 2. launch terminal and run "reboot" 	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1757: adjust date and timeSummary:

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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1758: switch among multi applications and desktopSummary:

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:	Fullpass
Case Automation Type:	Manual
Case State:	Ready

Feature:	system usage
target:	e-menlow, blacksand, beagleboard, crownbay, sugarbay, jasperforest
image profile:	sato, sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1759: vncserver for target	
<u>Summary:</u>	
Check if vncserver setup work in target and vnc client could connect it	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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" 	
<u>Expected Results:</u>	
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, qemumips, e-menlow, blacksand, crownbay, sugarbay, jasperforest
image profile:	sato, sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1760: file manager	
<u>Summary:</u>	
file manager	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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	
<u>Expected Results:</u>	
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

image profile:	sato, sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1761: system dmesg log check	
<u>Summary:</u>	
check if there is error in dmesg after system boot up	
<u>Steps:</u>	
1.make sure no other operation after startup the system. 2.run "dmesg grep -i error" in the terminal. 3.check if there is any error log printed.	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1762: usb mount	
<u>Summary:</u>	
verify that system can mount plugged usb automatically	
<u>Steps:</u>	
1. boot system 2. plug usb stick	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1763: usb read files	
<u>Summary:</u>	
verify that system can read files from usb	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. boot system 2. plug usb stick 3. view files in usb by file browser 4. copy some files from usb to local hardware 	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1764: usb unmount	
<u>Summary:</u>	
verify that system can unmount usb automatically	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. boot system 2. plug usb stick 3. view files in usb by file browser 4. unplug usb 	
<u>Expected Results:</u>	
1. usb directory 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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1765: usb write files	
<u>Summary:</u>	
verify that system can write files to usb	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. boot system 2. plug usb stick 3. create files in usb 4. copy some files from local hardware to usb 	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1766: file copy by scp	
<u>Summary:</u>	
check if file can be copied from remote machine to device by scp	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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) 	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1767: connman launch after boot
<u>Summary:</u>

After system booted, the connmand daemon should be launched	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. boot system 2. "ps aux grep connmand" or "ps -ef grep connmand" 3. check if there is a thread named connmand in background 	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1768: ethernet enabled in connman	
<u>Summary:</u>	
After system boot, ethernet can get IP address with connman	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1769: only one connmand in background	
<u>Summary:</u>	
there should be no more than one connmand in background	
<u>Steps:</u>	

<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	
There will be only one connmand instance in background	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	connectivity
target:	qemuarm, qemuppc, qemumips, e-menlow, blacksand, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest
image profile:	sato, sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1770: remote access by ssh	
<u>Summary:</u>	
check if the device can be accessed remotely by ssh	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. check dropbear is install and started 2. get system IP and try "ssh \$IP" from remote machine 	
<u>Expected Results:</u>	
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:	qemuarm, qemuppc, qemumips, e-menlow, blacksand, mpc8315e-rdb, routerstationpro, crownbay, sugarbay, jasperforest
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1771: ethernet static ip set in connman	
<u>Summary:</u>	
we could set static ip for ethernet in connman	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. launch connman-properties 	

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:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	connectivity
target:	e-menlow, blacksand, crownbay, sugarbay, jasperforest
image profile:	sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1772: 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:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	connectivity
target:	e-menlow, blacksand, crownbay, sugarbay, jasperforest
image profile:	sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1773: connman offline mode in connman-gnome

Summary:

change offline mode in connman-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	
<u>Expected Results:</u>	
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, qemumips, e-menlow, blacksand, crownbay, sugarbay, jasperforest
image profile:	sato, sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1774: X server can start up with runlevel 5 boot	
<u>Summary:</u>	
check if X server can work well after system runlevel 5 booting	
<u>Steps:</u>	
1. boot up system with default runlevel	
<u>Expected Results:</u>	
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, qemumips, e-menlow, blacksand, beagleboard, crownbay, sugarbay, jasperforest
image profile:	sato, sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1775: qt application quicky	
<u>Summary:</u>	
quicky is a simple note-taking application with Wiki-style syntax and behaviour	
<u>Steps:</u>	
launch quicky and write something in quicky	
<u>Expected Results:</u>	
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
image profile:	sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1776: 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
image profile:	sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1777: check CPU utilization after standby

Summary:

check CPU utilization after standby	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	
There should be no big difference before/after standby with "top"	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	crownbay, sugarbay
image profile:	sato, sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1778: Test if LAN device works well after resume from suspend state	
<u>Summary:</u>	
Test if LAN device works well after resume from suspend state.	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	
ping should always work before/after standby	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	e-menlow, blacksand, crownbay, sugarbay, jasperforest
image profile:	sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1779: Test if usb hid device works well after resume from suspend state	
<u>Summary:</u>	
Test if usb hid device works well after resume from suspend state.	
<u>Steps:</u>	

<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	
usb mouse and keyboard should work	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	system usage
target:	e-menlow, blacksand, crownbay, sugarbay, jasperforest
image profile:	sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1780: disk space check	
<u>Summary:</u>	
There should be enough disk space for QEMU rootfs	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	
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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1781: click terminal icon on X desktop	
<u>Summary:</u>	
terminal icon should work without problem on X desktop	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	

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, qemumips, e-menlow, blacksand, beagleboard, crownbay, sugarbay
image profile:	sato, sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1782: Add multiple files in music player	
<u>Summary:</u>	
music player should be no problem when adding multiple files at same time	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. Launch music player 2. Add multiple files(5 files) in music player at same time 	
<u>Expected Results:</u>	
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
image profile:	sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1783: system shutdown with UNFS	
<u>Summary:</u>	
system shutdown with UNFS should work	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. Use UNFS to start QEMU targets 2. Run shutdown in QEMU targets 	
<u>Expected Results:</u>	
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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1784: no connman-gnome icon on desktop	
<u>Summary:</u>	
there should be no connman-gnome icon on desktop	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. Launch sato image 2. There should be no connman-gnome icon on desktop, and connman-properties should be only invoked by toolbar 	
<u>Expected Results:</u>	
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, qemumips, e-menlow, blacksand, beagleboard, crownbay, sugarbay
image profile:	sato, sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1785: application contacts should work	
<u>Summary:</u>	
application contacts should work without problem	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	
"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, qemumips, e-menlow, blacksand, beagleboard, crownbay, sugarbay

image profile:	sato, sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1786: x11vnc icon click for target	
<u>Summary:</u>	
Check if vncserver could work in target by clicking x11vnc icon	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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" 	
<u>Expected Results:</u>	
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, qemumips, e-menlow, blacksand, crownbay, sugarbay
image profile:	sato, sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1787: RTLDLIST path check for ldd command	
<u>Summary:</u>	
check if the file set in RTLDLIST is valid	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. After system is up, check if the RTLDLIST variable in ldd command 2. The file path set in RTLDLIST should be valid 	
<u>Expected Results:</u>	
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
image profile:	sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1788: check bash in image	
<u>Summary:</u>	
check if bash exists in image	
<u>Steps:</u>	
1. After system is up, check if bash command exists with command "which bash"	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1789: "Install/Remove Software" icon should be removed	
<u>Summary:</u>	
"Install/Remove Software" icon should be removed from sato	
<u>Steps:</u>	
1. After system is up, there should be no "Install/Remove Software" icon	
<u>Expected Results:</u>	
"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, qemumips
image profile:	sato, sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

1.3 Test Suite : ADT

Test Case TC-1790: gcc from ADT toolchain can build c programSummary:

gcc from ADT 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 -cc -lm"
3. run "test" with qemu- $\{ARCH\}$ or run it into corresponding target image and check the output

Note: Currently, only i586_x86-64, x86-64_x586 and i586_\$(x is mips, arm and ppc) toolchain tarballs are covered in testing.

```
#####
#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("floor(%f) = %f\n", f, floor(f));
    return 0;
}
#####
```

Expected Results:

executable binary test can run without problem

Test Execution Cycle Type:	Sanity
Case Automation Type:	Auto
Case State:	Ready
Feature:	sdk
target:	build_system
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1791: g++ from ADT toolchain can build c programSummary:

g++ from ADT 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 -cc++ -lm"
3. run "test" with qemu-\${ARCH} or run it in corresponding target image and check the output

Note: Currently, only i586_x86-64, x86-64_i586 and i586_\$(x is mips, arm and ppc) toolchain tarballs are covered in testing.

```
#####  
#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:	Sanity
Case Automation Type:	Auto
Case State:	Ready
Feature:	sdk
target:	build_system
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1792: ADT toolchain could build cvs project

Summary:

ADT 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"	
Note: Currently, only i586_i586, x86-64_x86-64 and i586_\$X(x is mips, arm and ppc) toolchain tarballs are covered in testing.	
<u>Expected Results:</u>	
cvs project could be compiled successfully with ADT toolchain	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	build_system
image profile:	Isb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1793: ADT toolchain could build iptables project	
<u>Summary:</u>	
iptables project could be compiled with ADT toolchain	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. Install toolchain tarball and setup cross compile environment 2. Download iptables project, http://netfilter.org/projects/iptables/files/iptables-1.4.11.tar.bz2 3. With the cross compile environment, run ".configure \${CONFIGURE_FLAGS}", "make", "make install DESTDIR=/opt/tmp" 	
Note: Currently, only i586_i586, x86-64_x86-64 and i586_\$X(x is mips, arm and ppc) toolchain tarballs are covered in testing.	
<u>Expected Results:</u>	
iptables could be compiled successfully	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	build_system
image profile:	Isb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1794: ADT toolchain could build sudoku-savant project	
<u>Summary:</u>	
sudoku-savant could be compiled with ADT toolchain	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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}`, `make`, `make install DESTDIR=/opt/tmp`

Note: Currently, only i586_i586, x86-64_x86-64 and i586_\$(x is mips, arm and ppc) toolchain tarballs are covered in testing.

Expected Results:

sudoku-savant could be compiled successfully

Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	sdk
target:	build_system
image profile:	lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1795: unfs support for qemu target

Summary:

Check if unfs works for qemu target

Steps:

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

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, sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

1.4 Test Suite : Stress

Test Case TC-1796: crashme for stress

Summary:

Run crashme in real hardware for stress testing	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	
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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1797: helltest for stress	
<u>Summary:</u>	
Run helltest for stress in target	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. helltest is stress test suite, which does compiler test for hours 2. We download the test suite and run it for 24 hours 	
<u>Expected Results:</u>	
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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1798: ltp for stress	
<u>Summary:</u>	
Run ltp stress in real hardware for stress testing	
<u>Steps:</u>	
LTP download: http://sourceforge.net/projects/ltp/files/LTP%20Source/ltp-20101031/ltp-full-20101031.bz2/download	

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	
<u>Expected Results:</u>	
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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

1.5 Test Suite : Power/Performance

Test Case TC-1799: boot time collection	
<u>Summary:</u>	
To collect boot time of clean installation, from grub to full desktop	
<u>Steps:</u>	
1. Reboot testing device at least 3 times and do not plug anything while collecting boot time by stopwacher:	
#reboot	
<u>Expected Results:</u>	
Provide average boot time and dmesg log	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	performance
target:	crownbay, sugarbay
image profile:	sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1800: memory footprint	
<u>Summary:</u>	
collect data of the used/free memory	
<u>Steps:</u>	
With default installtion, launch terminal and type 'free' to read the used/free disk space	
<u>Expected Results:</u>	
Provide 'free' output	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	core
target:	crownbay, sugarbay
image profile:	sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1801: powertop log	
<u>Summary:</u>	
collect powertop data	
<u>Steps:</u>	
1. Run "powertop -d" and record output	
2. Save the percentage of deepest C state(C3 or C2)	
<u>Expected Results:</u>	
Provide powertop output	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	core
target:	crownbay, sugarbay
image profile:	sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1802: Idle power consumption	
<u>Summary:</u>	
Collect idle power consumption of target system	
<u>Steps:</u>	

1. Use power meter to collect ilde power consumption of target system for 10 minutes	
2. Save it and compare it with old data	
<u>Expected Results:</u>	
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, sugarbay
image profile:	sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1803: core build time for sato image	
<u>Summary:</u>	
collect the core build time for sato qemu86 image	
<u>Steps:</u>	
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 qemu86 sato image and collect the time	
<u>Expected Results:</u>	
There should be no regression for build time	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	performance
target:	qemu86_32
image profile:	sato
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

1.6 Test Suite : Graphics

Test Case TC-1804: 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 @@
        #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
```

<pre> 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 ##### </pre>	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1805: openarena - 3D	
<u>Summary:</u>	
Run openarena testing and compare the result with upstream graphics result	
<u>Steps:</u>	
<p>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.</p> <pre> #### phoronix-test-suite list-tests phoronix-test-suite install openarena #### </pre> <p>2.Go into the directory of openarena sourcecode folder.</p> <p>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.</p> <p>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".</p> <p>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.</p> <p>6.Run the test suite with following command:</p> <pre> vblank_mode=0 ./openarena +exec pts +set r_mode -1 +set r_fullscreen 1 +set r_customWidth \$VIDEO_WIDTH +set r_customHeight \$VIDEO_HEIGHT </pre> <p>The VIDEO_WIDTH and VIDEO_HEIGHT set the game's resolution, you can get current resolution by command "xrandr"</p>	
<u>Expected Results:</u>	
Compare the result of Yocto with upstream graphics	
Test Execution Cycle Type:	Weekly
Case Automation Type:	Manual
Case State:	Ready
Feature:	bsp
target:	sugarbay

image profile:	sato, sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1806: urbanterror - 3D	
<u>Summary:</u>	
Run urbanterror and compare the result of Yocto with upstream graphics	
<u>Steps:</u>	
<p>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 ###</p>	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1807: x11perf - 2D	
<u>Summary:</u>	
Get fps data of x11per running	
<u>Steps:</u>	
<p>1. Run "x11perf -aa10text" and "x11perf -rgb10text" 2. Get the FPS result and compare it with upstream graphics data on Sandybridge</p>	
<u>Expected Results:</u>	
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
image profile:	sato, sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

1.7 Test Suite : Multimedia

Test Case TC-1808: libva check (ogg video play)	
<u>Summary:</u>	
check if libva is installed and used when video player playing ogg video file	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. check if libva is installed on system 2. copy sample ogg file to system 3. launch video player can play the ogg file 	
<u>Expected Results:</u>	
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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1809: sound on/off	
<u>Summary:</u>	
check if sound can be turned on/off	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	
Above commands can run without problem	
Test Execution	Weekly

Cycle Type:	
Case Automation Type:	Manual
Case State:	Ready
Feature:	multi-media
target:	e-menlow, blacksand, beagleboard, crownbay, sugarbay
image profile:	sato-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1810: audio play (mp3)	
<u>Summary:</u>	
make sure music player cannot play mp3 format file	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. copy sample mp3 file to system 2. launch music player and make sure it cannot play the mp3 file 	
<u>Expected Results:</u>	
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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1811: audio play (ogg)	
<u>Summary:</u>	
check if music player can play ogg format file	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. copy sample ogg file to system 2. launch music player can play the ogg file 	
<u>Expected Results:</u>	
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

<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1812: audio stop (ogg)	
<u>Summary:</u>	
check if music player can play ogg format file	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	
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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1813: audio play (wav)	
<u>Summary:</u>	
check if music player can play wav format file	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. copy sample wav file to system 2. launch music player can play the wav file 	
<u>Expected Results:</u>	
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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1814: audio stop (wav)	
<u>Summary:</u>	
check if music player can stop playing with wav format file	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	
wav 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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1815: video play (mpeg)	
<u>Summary:</u>	
make sure video player cannot play mpeg format file	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. copy sample mpeg file to system 2. launch video player and make sure it cannot play the mpeg file 	
<u>Expected Results:</u>	
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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1816: video play (ogg)	
<u>Summary:</u>	
check if video player can play ogg format file	
<u>Steps:</u>	

1. copy sample ogg file to system 2. launch video player can play the ogg file	
<u>Expected Results:</u>	
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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1817: video stop (ogg)	
<u>Summary:</u>	
check if video player can play ogg format file	
<u>Steps:</u>	
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	
<u>Expected Results:</u>	
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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

1.8 Test Suite : Compliance

Test Case TC-1818: LTP subset test suite
<u>Summary:</u>
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://downloads.sourceforge.net/project/ltp/LTP%20Source/ltp-20110915/ltp-full-20110915.bz2?r=&ts=1325052583&use_mirror=ncu

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
<u>Last Result</u>	Not Run

<u>Keywords:</u>	None
------------------	------

Test Case TC-1819: 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 command: make generate-makefiles
4. Run command: make conformance-all
5. Run command: make conformance-test (this step may)
6. Run command: make tools-all
7. Run command: 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, qemuarm, qemumips, blacksand, beagleboard, mpc8315e-rdb, routerstationpro, sugarbay
image profile:	sato-sdk, lsb-sdk
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1820: 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	
<u>Expected Results:</u>	
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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

1.9 Test Suite : Core Build System

Test Case TC-1828: Init scripts	
<u>Summary:</u>	
Provide an image/recipe skeleton as a canonical example. Check if can be built and run correctly	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. Build image from poky source, check if skeleton script and skeleton-test can be built into the image <ol style="list-style-type: none"> a. download poky source b. modify the line <code>IMAGE_FEATURES += "apps-console-core \${SATO_IMAGE_FEATURES}"</code> to <code>IMAGE_FEATURES += "apps-console-core \${SATO_IMAGE_FEATURES}} service"</code> in <code>meta/recipes-sato/images/core-image-sato.bb</code> (for sato image) or <code>core-image-sato-sdk.bb</code> (for sato-sdk image) c. <code>\$ source oe-init-build-env</code> add line "<code><POKY_BASE>/meta-skeleton \</code>" to <code>conf/bblayer.conf</code> d. build the image e. boot up the image, check the skeleton and skeleton-test should be in right place <code>/etc/init.d/skeleton</code> <code>/usr/sbin/skeleton-test</code> 2. Verify the basic function of skeleton. Check if skeleton script can start/stop the skeleton-test daemon. 	
<u>Expected Results:</u>	
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:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1830: 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:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1831: 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 ccache-native -c clean
bitbake less -c clean
bitbake less -c compile
```

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

3. Compile less bbfile again with native ccache support

```
bitbake less -c clean
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-native 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:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1832: 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:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1853: Gtk+ Over DirectFBSummary:

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 arpg bluetooth ext2 irda largefile pcmcia usb gadget 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:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1854: bitbake-runtaskSummary:

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:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1855: 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:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1874: Archive work directory

Summary:

Archives up the work directories at the end of each build

Steps:

1. Download poky source and prepare the build environment
2. Set "inhert archive" to meta/recipes-extended/man/man_1.6f.bb
Run "bitbake man" to build man package.
3. Set "inhert archive" to /meta/classes/base.bbclass
Run "bitbake core-image-sato" to build image.

Expected Results:

After step 2, the source tarball for man should be in "\$TMPDIR/archives/\${MULTIMACH_TARGET_SYS}/"
After setp 3, the source tarball for all package should be in "\$TMPDIR/archives/\${MULTIMACH_TARGET_SYS}/"

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

<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1876: Disk space monitoring	
<u>Summary:</u>	
Monitor disk availability and warn the user if it is running low	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. Download the source and set build environment. 2. Add the following lines to conf/local.conf: <pre># Set the directories to monitor for disk usage BB_DISKMON_DIRS = "\${TMPDIR} \${DL_DIR} \${SSTATE_DIR}" # Set the minimum amount of disk space to warn BB_DISKMON_MINSPACE = "1GB" # Set the minimum inodes of disk space to warn BB_DISKMON_MININODES = "20M" # Set the monitor frequency BB_DISKMON_INTERVAL = "60s"</pre> 3. Run "bitbake core-image-sato" to build a image. 	
<u>Expected Results:</u>	
<p>Running "df -h " or "df -i" to check the free space or free inodes of the disk. When the free disk space or free inodes less than the setting values, a warning message will be printed every "BB_DISKMON_INTERVAL" time. The warning format is similar to: "WARNING: the space of /buildarea/test/downloads is running low (1GB left)".</p> <p>Press "Ctrl + C" to terminate the build process and the warning printing should also be stopped.</p>	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1877: Incremental image generation	
<u>Summary:</u>	
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".	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. Download poky source and prepare the build environment 2. Add the following line to conf/local.conf: <pre>INC_IMAGE_GEN = "1"</pre> 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. 	
<u>Expected Results:</u>	
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:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1878: Export source package from work	
<u>Summary:</u>	
Export the patches sources in a tarball or package format to assist in license compliance by the end user.	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. Download poky source and prepare the build environment 2. Set "inherit tar_archive" to meta/classes/package_rpm.bbclass 3. Set "ARCHIVE_TYPE ?= "SRPM"" to conf/local.conf Run "bitbake core-image-sato -c copysources" 4. Modify the "ARCHIVE_TYPE ?= "SRPM"" to "ARCHIVE_TYPE ?= "TARGZ"" in conf/local.conf Run "bitbake core-image-sato -c copysources" 5. Set "inherit tar_archive" to a package.bb file, for example, zlib: Add "inherit tar_archive" to meta/recipes-core/zlib/zlib_1.2.5.bb Run "bitbake zlib" 	
<u>Expected Results:</u>	
<p>After step 3, the source rpm for all packages should be in "\${DEPLOY_DIR}/sources"</p> <p>After step 4, the source tarballs for all package should be in "\${DEPLOY_DIR}/sources"</p> <p>After step 5, the source tarball for zlib should be in "build/tmp/work/i586-poky-linux/zlib-1.2.5-r3/deploy-srpm/i586/", for qemu86 as an example.</p>	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1717: hob launch against self-hosted-image with QEMU	
<u>Summary:</u>	
check if self-hosted-image could launch hob with QEMU	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. Get poky source code and prepare the build environment 2. Set MACHINE to qemu86-64 and run "bitbake self-hosted-image" 3. After build is finished, start QEMU and setup poky build environment in the QEMU 4. Launch hob in QEMU 	
<u>Expected Results:</u>	
hob could be launched against self-hosted-image with QEMU	

Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1821: lib64 sato image build - qemu86-64/ipk	
<u>Summary:</u>	
lib64 sato image should be built out with multilib support	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 qemu86-64 as following: ##### MACHINE = "qemu86-64" require conf/multilib.conf MULTILIBS = "multilib:lib64" DEFAULTTUNE_virtclass-multilib-lib64 = "x86-64" ##### 3. with ipk set for package format, build lib64 core-sato image 4. after build finished, start up the image and check if all app are 64-bit, kernel with 64-bit 	
<u>Expected Results:</u>	
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:	build_system
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1822: lib64 sato image build - qemu86-64	
<u>Summary:</u>	
lib64 sato image should be built out with multilib support	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 qemu86-64 as following: ##### MACHINE = "qemu86-64" require conf/multilib.conf MULTILIBS = "multilib:lib64" DEFAULTTUNE_virtclass-multilib-lib64 = "x86-64" ##### 	

3. with rpm set for package format, build lib64 core-sato image	
4. after build finished, start up the image and check if all app are 64-bit, kernel with 64-bit	
<u>Expected Results:</u>	
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:	build_system
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1823: lib64 sato image build - qemu86	
<u>Summary:</u>	
lib64 sato image should be built out with multilib support	
<u>Steps:</u>	
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 qemu86 as following: ##### MACHINE = "qemu86" require conf/multilib.conf MULTILIBS = "multilib:lib64" DEFAULTTUNE_virtclass-multilib-lib64 = "x86-64" ##### 3. with rpm set for package format, build lib64 core-sato image 4. after build finished, start up the image and check if all app are 64-bit, kernel with 32-bit	
<u>Expected Results:</u>	
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:	build_system
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1718: bitbake fetch against self-hosted-image with QEMU	
<u>Summary:</u>	
check if bitbake fetch could work against self-hosted-image with QEMU	
<u>Steps:</u>	

<ol style="list-style-type: none"> 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 QEMU and setup poky build environment in the QEMU, 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 	
<u>Expected Results:</u>	
bitbake fetch could work against self-hosted-image with QEMU	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1824: lib32 sato image build - qemux86-64	
<u>Summary:</u>	
lib32 sato image should be built out with multilib support	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 image 4. after build finished, start up the image and the kernel should not be able to boot up 	
<u>Expected Results:</u>	
lib32 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:	build_system
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1716: minimal build with self-hosted-image with QEMU	
<u>Summary:</u>	
check if self-hosted-image could pass minimal build with QEMU	

<u>Steps:</u>	
<ol style="list-style-type: none"> 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 QEMU and setup poky build environment in the QEMU 4. Build a minimal image in the QEMU 	
<u>Expected Results:</u>	
self-hosted-image could pass minimal build with QEMU	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1825: lib32 sato image build - qemux86	
<u>Summary:</u>	
lib32 sato image should be built out with multilib support	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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: <pre>##### MACHINE = "qemux86" require conf/multilib.conf MULTILIBS = "multilib:lib32" DEFAULTTUNE_virtclass-multilib-lib32 = "x86" #####</pre> 3. with rpm set for package format, build lib32 core-sato image 4. after build finished, start up the image and check if all app are 32-bit, kernel with 32-bit 	
<u>Expected Results:</u>	
lib32 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:	build_system
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1826: lib32 connman-gnome built for qemux86-64 - rpm	
<u>Summary:</u>	
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 lib64 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:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	core
target:	
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1827: 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 lib64 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:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	core
target:	

image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1833: kernel interactive targets	
<u>Summary:</u>	
Check if yocto can support kernel interactive target build	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	
menuconfig for kernel can be triggered with yocto build command	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1834: KVM enabled with qemu	
<u>Summary:</u>	
qemu can be started with KVM enabled	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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" 	
<u>Expected Results:</u>	
KVM enabled with qemu	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1835: non-GPLv3 build checkSummary:

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:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1836: yocto build in Fedora 15Summary:

Build latest yocto in x86_64 Fedora 15 host

Steps:

1. By following the yocto handbook, download latest yocto source 2. Build core-image-minimal on Fedora 15	
<u>Expected Results:</u>	
Yocto build should pass on Fedora 15	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1837: yocto build in OpenSuse 11.4	
<u>Summary:</u>	
Build latest yocto in x86_64 OpenSuse 11.4	
<u>Steps:</u>	
1. By following the yocto handbook, download latest yocto source 2. Build core-image-minimal on OpenSuse 11.4	
<u>Expected Results:</u>	
Build should pass on OpenSuse 11.3	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1838: yocto build in Ubuntu 11.04	
<u>Summary:</u>	
Build latest yocto in x86_64 Ubuntu 11.04	
<u>Steps:</u>	
1. By following the yocto handbook, download latest yocto source 2. Build core-image-minimal on Ubuntu 11.04	
<u>Expected Results:</u>	
Yocto build should pass on Ubuntu 10.04	
Test Execution	Fullpass

Cycle Type:	
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1839: yocto build in KVM	
<u>Summary:</u>	
Build yocto in KVM should work	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	
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:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1840: sstate work on local host	
<u>Summary:</u>	
Check if sstate could work with local cache	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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)	

#####	
<u>Expected Results:</u>	
sstate should work and reduce build time	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1841: gcc set to 4.5.1 for core build	
<u>Summary:</u>	
gcc related options should be set to 4.5.1 for 4.5.1 build	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. Download poky source and prepare the build environment 2. Set GCCVERSION and SDKGCCVERSION to 4.5.1 in meta/conf/distro/include/tcmode-default.inc 3. Run "bitbake -s grep gcc" and check the output, all gcc related options should be set to 4.5.1 	
<u>Expected Results:</u>	
all gcc related options should be set to 4.5.1	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1842: btrfs format image build	
<u>Summary:</u>	
btrfs format image could be built out	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	
btrfs format image could be built out	
Test Execution Cycle Type:	Fullpass

Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1843: btrfs format image boot up	
<u>Summary:</u>	
btrfs format image could be booted up	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. set IMAGE_FSTYPES = "btrfs" and KERNEL_FEATURES_append = " cfg/btrfs " in local.conf 2. build a qemu86 core-image-minimal image and boot up it 	
<u>Expected Results:</u>	
btrfs format image could be booted up	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1844: bitbake-layers show_layers	
<u>Summary:</u>	
show_layers could show current layers	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	
show_layers could show current layers	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	

<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1845: bitbake-layers show_overlayed	
<u>Summary:</u>	
overlayed recipes should be shown with bitbake-layers	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	
overlayed recipes should be shown with bitbake-layers	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1846: bitbake-layers show_appends	
<u>Summary:</u>	
bitbake-layers show_appends should list bbappend files and recipe files they apply to	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. prepare poky build environment 2. run "bitbake-layers show_appends", it should list bbappend files and recipe files they apply to 	
<u>Expected Results:</u>	
bitbake-layers show_appends should list bbappend files and recipe files they apply to	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1847: bitbake-layers flattenSummary:

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:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
target:	build_system
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1848: buildhistory enable for yocto buildSummary:

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" in local.conf).
2. Once a build with package history enabled has finished, verify that the output can be found in TMPDIR/pkghistory.

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:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1849: buildhistory error if do_package backwards

<u>Summary:</u>	
check if buildhistory reports error if PR of some recipes go backwards	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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 	
<u>Expected Results:</u>	
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:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1851: x32 image build	
<u>Summary:</u>	
x32 image could be built out successfully	
<u>Steps:</u>	
<ol style="list-style-type: none"> 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" 	
<u>Expected Results:</u>	
x32 image could be built out successfully	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	core
target:	
image profile:	
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1852: x32 image build boot up and check	
<u>Summary:</u>	
x32 image could be built out successfully and binaries/libraries are x32 in it	
<u>Steps:</u>	

1. Prepare yocto build environment
 2. add meta-x32 layer, <http://git.yoctoproject.org/cgi/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:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	core
target:	
image profile:	
Last Result	Not Run
Keywords:	None

Test Case TC-1865: 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" 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
Last Result	Not Run
Keywords:	None

Test Case TC-1860: 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\_package 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           |
| <u>Last Result</u>         | <b>Not Run</b> |
| <u>Keywords:</u>           | None           |

### Test Case TC-1861: 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_package 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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1862: 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
3. run "bitbake man"
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_package for recipe man should be re-run
8. check the stamps folder, the do_package file for man should be regenerated 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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1863: 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
3. run "bitbake man"
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"
```

<pre> } ##### 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 </pre>	
<u>Expected Results:</u>	
check if AUTOPR could be export/lockdown for package build	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
Last Result	Not Run
Keywords:	None

Test Case TC-1864: AUTOPR export/import	
<u>Summary:</u>	
check if AUTOPR could be export/import for package build	
<u>Steps:</u>	
<pre> 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 </pre>	
<u>Expected Results:</u>	
check if AUTOPR could be export/import for package build	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
Last Result	Not Run
Keywords:	None

Test Case TC-1872: emgd driver auto download	
<u>Summary:</u>	
check if latest emgd driver could be downloaded automatically without manual copy	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. prepare poky and meta-intel build environment 2. set MACHINE to crownbay and include meta-intel into bblayers 3. set LICENSE_FLAGS_WHITELIST = "License_emgd-driver-bin_1.10" in local.conf 4. build out emgd-driver and check if it needs manual download 	
<u>Expected Results:</u>	
latest emgd driver could be downloaded automatically without manual copy	
Test Execution Cycle Type:	Fullpass
Case Automation Type:	Manual
Case State:	Ready
Feature:	poky
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

1.10 Test Suite : BSP specific

Test Case TC-1856: RTC	
<u>Summary:</u>	
Check if RTC(Real Time Clock) can work correctly	
<u>Steps:</u>	
<ol style="list-style-type: none"> 1. Read time from RTC registers. <pre>root@localhost:/root> hwclock -r</pre> <p>Sun Mar 22 04:05:47 1970 -0.001948 seconds</p> 2. Set system current time <pre>root@localhost:/root> date 062309452008</pre> 3. Synchronize the system current time to RTC registers <pre>root@localhost:/root> hwclock -w</pre> 4. Read time from RTC registers <pre>root@localhost:/root> hwclock -r</pre> 5. Reboot target and read time from RTC again. 	
<u>Expected Results:</u>	

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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1857: Watchdog	
<u>Summary:</u>	
Check if watchdog can reset the target system	
<u>Steps:</u>	
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.	
<u>Expected Results:</u>	
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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1858: SATA	
<u>Summary:</u>	
Test general use of SATA device on target, like mount, umount, read and write.	
<u>Steps:</u>	
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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None

Test Case TC-1859: 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
<u>Last Result</u>	Not Run
<u>Keywords:</u>	None