LynxSecure 6.3.0 Getting Started and Configuration Guide

LynxSecure Release 6.3.0-rev16326

Contents

PREFACE .... **xiii**

About this Guide xiii
Intended Audience xiii
For More Information xiii
Typographical Conventions xiv
Special Notes xv
Technical Support xv
  Lynx Software Technologies, Inc. U.S. Headquarters xv
  Lynx Software Technologies, Inc. Europe xv

CHAPTER 1 LYNXSECURE DEVELOPMENT WORKFLOW **1**

Installation, Configuration, Deployment, and Maintenance Outline 1

CHAPTER 2 SETTING UP A LYNXSECURE DEVELOPMENT HOST **3**

Select Development or Evaluation Environment 3
Development Host Requirements 3
Installing the LynxSecure Development Tools 4
Obtaining a License 4
  Getting the FlexNet Host Information from Linux 4
  Setting up the License Files 5
Multi-User Environment and Multiple Projects 5
Development Host Services Configuration 6
  Prerequisites and Assumptions 6
  Network Configuration 7
  Enabling TFTP 7
  Configuring and Running the DHCP Server 7
  PXELINUX Configuration 8
  NFS Server Configuration 9
Booting the Hardware Discovery Linux from a CD/DVD-ROM or USB Storage (only x86 platform) 9

CHAPTER 3 SETTING UP THE TARGET SYSTEM **11**

Setting up x86 Targets 11
  Creating Partitions on the Storage Device 11
Setting up Xilinx Zynq Ultrascale+ MPSoC Targets 12
Setting up NXP S32V234 MPSoC Targets 12

CHAPTER 4 INTRODUCTION TO THE AUTOCONFIG TOOL **15**

Working with Target System Information 15
  Collecting Target Information on x86 Platform 15
  Collecting Target Information on Arm Platform 16
  Displaying the Target Hardware Information 18
  Working with Target System Information on the Development Host 21
Creating an SRP 22
Creating the XML File (HCV) Without Creating an SRP 22

System-wide Options When Creating an SRP or HCV 23
Creating a Simple SRP Image 25

CHAPTER 5 BOOTING SRPS 27
Network Boot 27
Network Boot Using PXE (only x86 platform) 28
Network Boot Using U-Boot (only Arm platform) 28
Boot from a Mass Storage Device Using LynxSecure Bootstrap Loader (only x86 platform) 29
Installing to Hard Disk (only x86 platform) 29
Installing to USB Stick (only x86 platform) 30
Configure the LynxSecure Bootstrap Loader (only x86 platform) 30
Boot LynxSecure Using Syslinux Boot Loader (only x86 platform) 31
Boot LynxSecure Using UEFI (only x86 platform) 32
Preparing UEFI SRP Boot Disk 33
Preparing UEFI SRP Boot Using PXE 33
Boot LynxSecure Using kexec command on Linux (only x86 platform) 33
Boot LynxSecure from a SD card (only Arm platform) 34
Passing Options to RIF from a Custom Bootstrap Loader 34
Segmented Boot Using U-Boot (only Arm platform) 36
Overview 36
Normal Boot Process 36
Components of a Segmented Boot 36
Configuring an SRP 37
Configuring Boot Trailers 37
Configuring U-Boot 38
Execute In Place 38

CHAPTER 6 INSTALLING GUEST OPERATING SYSTEMS 41
Generic Installation 41
Guest OS Installation on a Dedicated Disk 41
Guest OS Installation on a Virtualized Disk 42
KDI Installation 42
Restart the System 42

CHAPTER 7 SUBJECTS 45
Generic Subject Autoconfig Syntax 46
FV Subject Autoconfig Syntax 51
PV Linux Subject Autoconfig Syntax (only x86 platform) 53
PV LynxOS Subject Autoconfig Syntax (only x86 platform) 54
LSA Autoconfig Syntax 54
IBIT Subject Autoconfig Syntax 55
CBIT Subject Autoconfig Syntax 55
Sample Configuration 56

CHAPTER 8 PHYSICAL DEVICE ASSIGNMENT TO SUBJECTS 57
Pseudo-devices 57
Using SR-IOV Devices 57
Device Assignment Limitations 58
Autoconfig Syntax 59
Sample Configurations 62

CHAPTER 9 VIRTUAL DEVICE SERVER (ONLY X86 PLATFORM) 65
Autoconfig Syntax 65
VDS Console 66
Serial Console 66

VGA Console 66
SSH 66
Virtual Device Server Tools 67
VDSCLI 67
Subject Monitor Administration Tool 69
Cache and Memory Bandwidth Monitoring Tool 73
VDS Root File System Customization and Booting 74
Virtual Device Server Overlays 74
Fully customized Virtual Device Server 74
Booting VDS from Disk 74

CHAPTER 10 VIRTUAL PCI-TO-LPC BRIDGE (ONLY X86 PLATFORM) 75
Features 75
Limitations 75
Autoconfig Syntax 75
Sample Configurations 76

CHAPTER 11 VIRTUAL ACPI (ONLY X86 PLATFORM) 77
Features 77
Limitations 77
Autoconfig Syntax 77

CHAPTER 12 VIRTUAL AUDIO (ONLY X86 PLATFORM) 79
Features 79
Limitations 79
Autoconfig Syntax 79

CHAPTER 13 VIRTUAL BLOCK DEVICE (ONLY X86 PLATFORM) 81
Features 81
Limitations 82
IDE Controller Emulation 82
AHCI Controller Emulation 82
Autoconfig Syntax 82
Sample Configurations 83

CHAPTER 14 VIRTUAL KEYBOARD VIDEO MOUSE (ONLY X86 PLATFORM) 85
Features 85
VGA Emulation 85
PS/2 Keyboard/Mouse Emulation 85
Performance 86
Limitations 86
List of Verified Graphics Cards 87
Autoconfig Syntax 87
The Virtual Keyboard/Video/Mouse Administration Tool Syntax 88
Switch the KVM focus Subject 88
Get the in-focus KVM Subject 88

CHAPTER 15 VIRTUAL NETWORK (ONLY X86 PLATFORM) 89
Features 89
Limitations 91
Autoconfig Syntax 91
Virtual Network Device Configuration String Syntax (HCV) 93
Sample Configuration 94

CHAPTER 16 VIRTUAL UART (ONLY X86 PLATFORM) 95
Features 95
Limitations 95
Autoconfig Syntax 95
Sample Configurations 96

CHAPTER 17 VIRTUAL SHARED MEMORY 97
Features 97
Limitations 97
Autoconfig Syntax 97
Sample Configuration 98

CHAPTER 18 VIRTUAL USB (ONLY X86 PLATFORM) 99
USB Emulation Rules 99
Features 101
Limitations 102
    Virtual EHCI Limitations 103
List of Verified USB Devices 104
Autoconfig Syntax 104
USB Administration Tool Syntax 105
    Add a USB Emulation Rule 105
    Remove a USB Emulation Rule 106
    List All USB Emulation Rules 106
    Debugging Level 107
    Display the Debugging Level 107
    List All USB Devices 107
    Display the USB Emulation Status 108
    Show Help 109
Sample Configuration 109

CHAPTER 19 LSA.STORE FULL DISK ENCRYPTION (ONLY X86 PLATFORM) 113
Features 113
Workflow Overview 114
Configuring the Target's TPM 1.2 116
Creating Key Material for Virtual Disks 117
Autoconfig Syntax 117
Creating Encrypted Disk Images on the Host 118
Importing/Exporting data from a Host System 118
Setting up the Key Management Tools in a Standalone Environment 119
Limitations 119

APPENDIX A EXAMPLE CONFIGURATIONS 121
Para-Virtualized Linux Configuration 121
    One PV Linux with Direct Device Assignment 121
    One PV Linux with Direct Device Assignment and Virtual Network 121
    One PV Linux with Direct Device Assignment and Virtual UART 122
    Multiple PV Linux Subjects with Virtual Devices 122
Para-Virtualized Linux and Fully Virtualized Subjects 122
LynxSecure Applications (LSA) 123
UEFI based FV Guests 123
LynxOS-178 2.2.4 124
    LynxOS-178 2.2.4 as FV Guest OS on Zynq UltraScale+ MPSoC 124

APPENDIX B AUTOCONFIG REFERENCE 125
Commands Reference 125
Commands 125
Options Reference 126
Global Options 126
Common Subject Options 132
FV Subject Options 134
PV Linux Subject Options 139
PV LSA Subject Options 142
PV LynxOS Subject Options 144
VDS Subject Options 146

APPENDIX C INTRODUCTION TO LYNXSECURE LICENSE MANAGEMENT 151
FLEXnet Overview 152
Installing Lynx License Management Software on Linux Host 153
Setting up the License Files with Dedicated License Server 153
Use a Local Copy of the License File. 153
Use a Shared Copy of the License File Using NFS. 153
Adding LynxSecure License to an Existing License Management Server 153
Using the Lynx License Management Software 154
Starting the FLEXnet License Management Server 154
Checking the Status of FLEXnet Server 154
Stopping the License Management Server 155
Working with the Lynx Session Manager 155

APPENDIX D CHANGES IN THE AUTOCONFIG TOOL COMMAND LINE INTERFACE 157
LynxSecure 6.0 Autoconfig Tool 157
LynxSecure 5.3 Autoconfig Tool 161

APPENDIX E GLOSSARY 167
Acronyms 172

List of Figures

1-1. LynxSecure Development Workflow 1
5-1. Network Boot 27
5-2. Booting SRP Using LynxSecure Bootstrap Loader 29
15-1. NAT 89
15-2. Bridging mode 90
15-3. Direct subject-to-subject connection 91
15-4. Example Virtual Network Topology 94
19-1. In-SRP Workflow 114
19-2. In-Disk Workflow 115

List of Tables

3-1. LynxSecure System Partitions 12
4-1. Platform Devices verified on Xilinx Zynq Ultrascale+ MPSoC 16
4-2. Platform Devices verified on NXP S32V234 MPSoC 17
7-1. Autoconfig Tool Subject Type Syntax 47
7-2. Autoconfig Tool Boot Region Syntax 48
7-3. Permissions 49
7-4. The debugging levels 52
7-5. Supported boot devices 52
8-1. Device Assignment Syntax 61
9-1. VDSCLI Commands 67
9-2. The SAV Logging Level Options 72
9-3. CMT Fields Information 73
14-1. The Virtual Keyboard/Video/Mouse System Shortcut Keys 86
14-2. The List of Verified Graphics Cards 87
15-1. Virtual Network Device Configuration Options (legacy) 92
15-2. Virtual Network Device Configuration Options 93
15-3. Standard Virtual Network Device Configuration Options 93
18-1. Emulation Rules Specification 100
18-2. Examples 101
18-3. The maximum values of the NDP for different USB controller types 102
18-4. Tested USB devices 104
18-5. Output formats 106
18-6. The USB emulation debugging levels 107
18-7. Output formats 108
18-8. Default output format 108
18-9. Output format 109
D-1. Changes in Autoconfig Tool Commands 157
D-2. Changes in Autoconfig Tool Global Options 157
D-3. Changes in Autoconfig Tool Subject-Specific Options 159
D-4. Changes in Autoconfig Tool Commands 161
D-5. Changes in Autoconfig Tool Global Options 162
D-6. Changes in Autoconfig Tool Subject-Specific Options 164

Preface

About this Guide

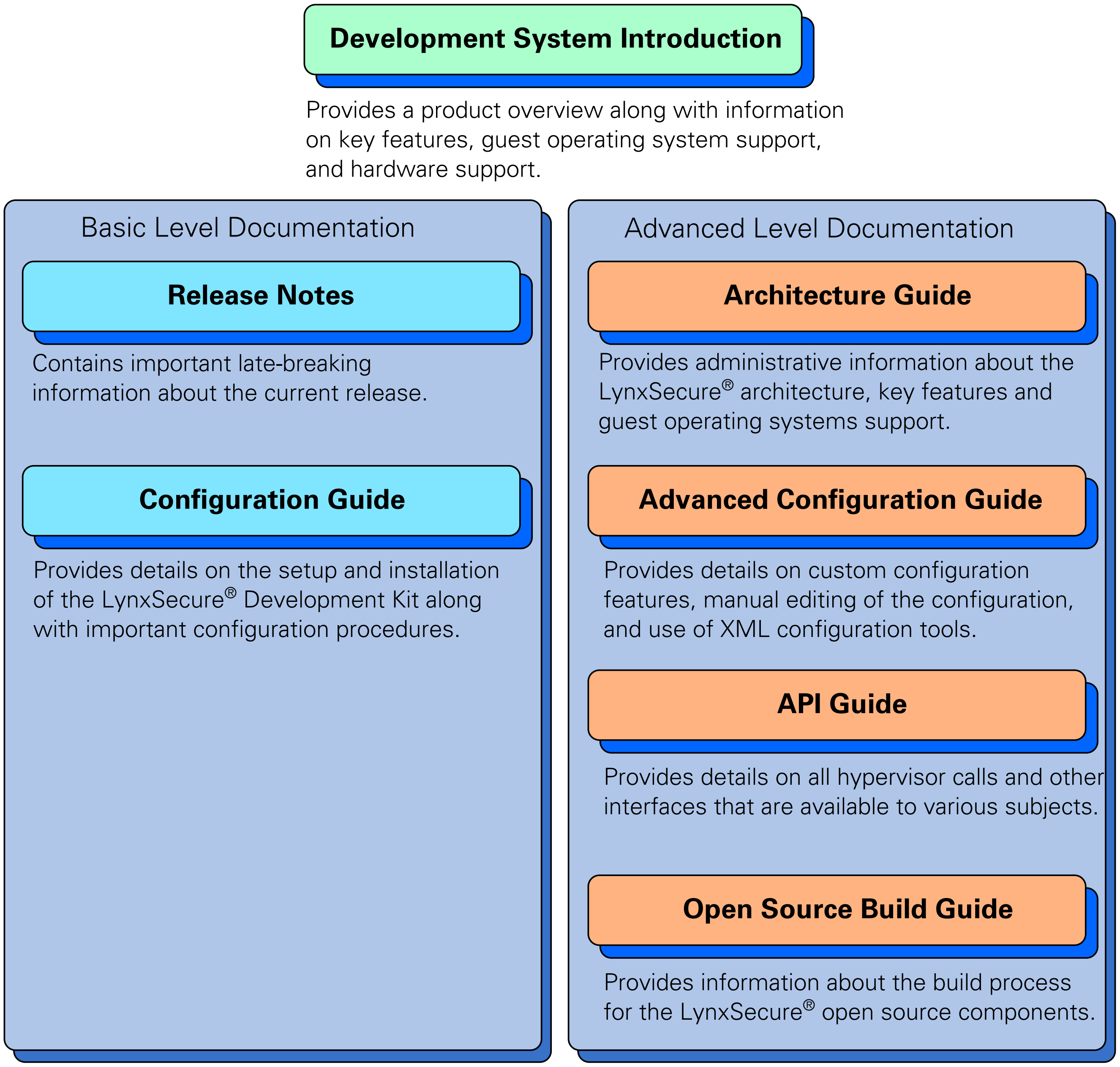
This guide, *LynxSecure® Getting Started and Configuration Guide*, provides information about installation and configuration procedures for LynxSecure and the subjects running on LynxSecure.

Intended Audience

The information in this guide is designed and written for system integrators and software developers who will build applications on top of LynxSecure and the available subjects. A basic understanding of Linux®/Unix® and familiarity with fairly complex configuration procedures are recommended.

For More Information

For more information on the features of LynxSecure, refer to the following printed and online documentation.


Development System Introduction
Provides a product overview along with information on key features, guest operating system support, and hardware support.
Basic Level Documentation
Release Notes
Contains important late-breaking information about the current release.
Configuration Guide
Provides details on the setup and installation of the LynxSecure® Development Kit along with important configuration procedures.
Advanced Level Documentation
Architecture Guide
Provides administrative information about the LynxSecure® architecture, key features and guest operating systems support.
Advanced Configuration Guide
Provides details on custom configuration features, manual editing of the configuration, and use of XML configuration tools.
API Guide
Provides details on all hypervisor calls and other interfaces that are available to various subjects.
Open Source Build Guide
Provides information about the build process for the LynxSecure® open source components.

Typographical Conventions

The typefaces used in this manual, summarized below, emphasize important concepts. All references to file names and commands are case sensitive and should be typed accurately.

Font and Description	Examples
Times New Roman 10 pt. - Used for body text; *italicized* for emphasis, new terms, and book titles.	Refer to the *LynxSecure User's Guide*

Courier New 9 pt. - Used for environment variables, file names, functions, methods, options, parameter names, path names, commands, and computer data. Commands that need to be	`ls -l`
	`myprog.c`
	`/dev/null`
	`login:` **`myname`**

Font and Description	Examples
highlighted within body text, or commands that must be typed as-is by the user are **bolded**.	`# `**`cd /usr/home`**

Courier New Italic 9 pt. - Used for text that represents a variable, such as a file name or a value that must be entered by the user.	`cat` *`filename`*
	`mv` *`file1 file2`*

Courier New 7 pt. - Used for blocks of text that appear on the display screen after entering instructions or commands.	`Kernel: target1.srp`
	`> Loading: target1.srp`
	`> .......................`
	`> Booting`

Univers 45 Light Bold 8 pt. - keyboard options, button names, and menu sequences.	Enter, Ctrl-C

Special Notes

The following notations highlight any key points and cautionary notes that may appear in this manual.

NOTE: These callouts note important or useful points in the text.

CAUTION! Used for situations that present minor hazards that may interfere with or threaten equipment/performance.

Technical Support

Lynx Software Technologies, Inc. Technical Support is available Monday through Friday (excluding holidays) between 8:00 AM and 5:00 PM Pacific Time (U.S. Headquarters) or between 9:00 AM and 6:00 PM Central European Time (Europe).

The Lynx Software Technologies, Inc. World Wide Web home page [http://www.lynx.com] provides additional information about our products.

Lynx Software Technologies, Inc. U.S. Headquarters

Internet: `<support@lynx.com>`
Phone: (408) 979-3940
Fax: (408) 979-3920

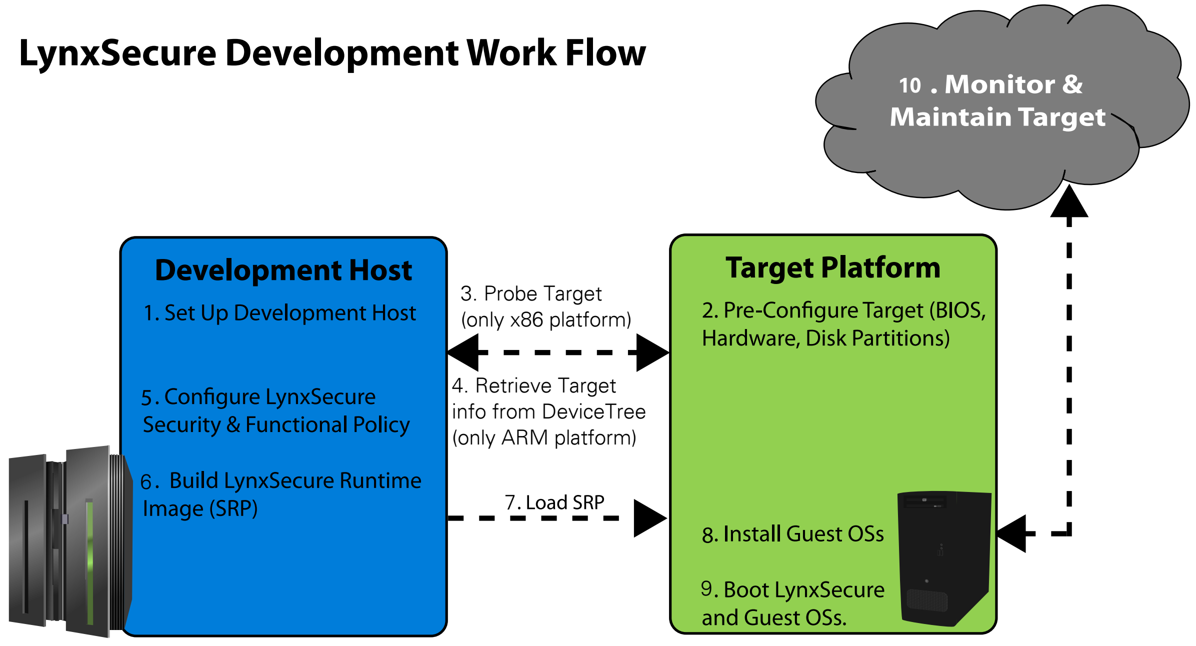
Lynx Software Technologies, Inc. Europe

Internet: `<tech_europe@lynx.com>`
Phone: (+33) 1 30 85 06 00
Fax: (+33) 1 30 85 06 06

CHAPTER 1 *LynxSecure Development Workflow*

This chapter illustrates standard start-to-finish workflow of using the LynxSecure® Development System.

Figure 1-1: LynxSecure Development Workflow


LynxSecure Development Work Flow
10 . Monitor & Maintain Target
Development Host
1. Set Up Development Host
5. Configure LynxSecure Security & Functional Policy
6. Build LynxSecure Runtime Image (SRP)
3. Probe Target (only x86 platform)
4. Retrieve Target info from DeviceTree (only ARM platform)
7. Load SRP
Target Platform
2. Pre-Configure Target (BIOS, Hardware, Disk Partitions)
8. Install Guest OSs
9. Boot LynxSecure and Guest OSs.

Installation, Configuration, Deployment, and Maintenance Outline

The following steps outline the process for installing, configuring, deploying, and maintaining the LynxSecure Development System:

1. Set up the Development Host (page 3)

   Install the LynxSecure development tools on the host development platform. The development tools run on a CentOS™ 7.3 or later operating system. The development tools also rely on several services that must be configured including a license server, a DHCP server, and a network boot server. To get started quickly, it is recommended that the host and the target be connected via Ethernet. The tools and services can be installed on a pre-existing customer owned installation of CentOS 7.3 or later, or can optionally be pre-installed with the purchase of a development host platform.

2. Set up the Target System (page 11)

Target systems running LynxSecure must be pre-configured to enable required features in the system firmware. For example, this step may include enabling VT-d support on x86 platforms.

3. Probe the Target (only applicable for a x86 target) (page 15)

   Once the target is pre-configured, users must then run a probe tool from the development host. This tool captures the required target hardware and configuration profile needed to start the configuration process.

4. Retrieve target information using a Device Tree (only applicable for a Arm target) (page 16)

   Once the target is pre-configured, users must then retrieve the target information using a command line tool. This tool captures the required target hardware and configuration profile needed to start the configuration process.

5. Configure LynxSecure (page 22)

   Next, a system configuration is generated using command line tools. This configuration is stored as XML and fully defines the run time configuration of the system while executing LynxSecure. This XML configuration is called the Human-readable Configuration Vector, and is often referred to as the HCV.

6. Build the LynxSecure Runtime Image (page 25)

   Using the HCV as an input, the configuration compiler generates the System Runtime Package. This binary contains the hypervisor along with any configured guest resources. The System Runtime Package is generally referred to as the SRP.

7. Load the SRP (LynxSecure System Runtime Package) (page 27)

   Once the SRP bootable image is generated by the configuration compiler, the bootable image must then be loaded onto the target platform. The SRP can be booted from many media formats including a Local Hard Disk, Flash Media, USB Media, or the network.

8. Install the Guest Operating System(s) (page 41)

   Now that the SRP is running on the target system, the guest operating systems can be configured and installed. While LynxSecure provides flexibility in this area, it is recommended that guest operating systems are installed from within a running instance of LynxSecure. This will ensure that all of the guest's resources are securely separated, and that no unauthorized resources will be configured during installation.

9. Boot LynxSecure & Guest Operating System(s) (page 42)

   Once the guest operating systems have been installed, LynxSecure must be fully rebooted by safely shutting down each guest and resetting the whole system.

10. Monitor and Maintain the Target

    Using the development tools, users have the option to connect to reference monitor agents included in the Virtual Device Server or to custom developed maintenance, security, and reliability software that may either be embedded alongside the guest operating system as autonomous components, or within the management guest Virtual Device Server.

CHAPTER 2 *Setting up a LynxSecure Development Host*

LynxSecure® Development Workflow Step 1

Install the LynxSecure development tools on the host development platform. The development tools run on a CentOS™ 7.3 or later operating system. The development tools also rely on several services that must be configured including a license server, a DHCP server, and a network boot server. To get started quickly, it is recommended that the host and the target be connected via Ethernet. The tools and services can be installed on a pre-existing customer owned installation of CentOS 7.3 or later, or can optionally be pre-installed with the purchase of a development host platform.

Select Development or Evaluation Environment

LynxSecure supports both an Evaluation Environment and a CDK environment using EVAL and CDK media, respectively. Note the key differences between the EVAL and CDK versions:

EVAL version:

- has no license management;
- will artificially slow down the guests after 8 hours of uptime;
- does not support LSA development.

CDK version:

- needs a license to work;
- creates deployable LynxSecure binaries (no timed slowdown);
- allows LSA development.

Development Host Requirements

- A machine capable of running 64-bit CentOS 7.3 or later
- A dedicated Ethernet port to boot the LynxSecure target using PXE

NOTE: It is recommended to have an Ethernet port dedicated for communication with target systems. If a dedicated port is not available extra care is required when configuring the DHCP service so that it does not interfere with the normal operation of your network. This configuration is not covered in this document.

Installing the LynxSecure Development Tools

LynxSecure requires a host machine running 64-bit CentOS 7.3 or later or later. Insert the provided DVD into your workstation, and execute the `install-lsk` script from the top level of the install media. This will install the RPMS containing the product to your workstation. Once this process is complete, the product will be available at `/opt/lynxsecure`.

If a full installation of CentOS is needed as well, there is a variant of the LynxSecure install media that contains a CentOS 7.3 or later installer. This installer will automatically install the LynxSecure RPMs as part of the normal install process. To use this method, insert the DVD into your workstation, select it as the boot device, and follow the on-screen prompts to configure your system.

Once the LynxSecure tools are installed to the system, host services will still need to be configured. Please refer to section "Development Host Services Configuration" (page 6) for more information.

Obtaining a License

The CDK version of LynxSecure environment requires a valid license to operate. For evaluation version, this section can be skipped. This section will describe the procedure to create the simplest development host set up (single machine running both as a license server and a development host, using node-locked license). Refer to Appendix C, "*Introduction to LynxSecure License Management*" (page 151) for additional information on license management for LynxSecure.

CAUTION! The host where the license server is to be running must have a host name which can be resolved to an IP address, using either DNS or the `/etc/hosts` file.

To obtain a license, please complete and submit the *Lynx Software Technologies License Key Request Form* that is provided with your product shipment. Once complete, the form can be emailed to `<license@lynx.com>` or faxed to (408) 979-3920.

You will be requested to supply the following information:

- Your contact information including a valid email address
- Product serial number
- Sales order number or purchase order number
- Product version and target platform
- The FlexNet host specific information

NOTE: For floating licenses, the user is requested to provide FlexNet host specific information for each machine where the license server (or servers) will run. For node-locked licenses, the user needs to provide FlexNet additional host specific information for each host where the Lynx products will be used.

In this simplest setup, the development host is the same machine as the license server.

Getting the FlexNet Host Information from Linux

The required host information includes the FlexNet host ID, Lynx composite host ID and host name. Perform the following steps:

1. Execute the `lwhostid` command:

```
# /opt/lw-flexlm/v11.14.1.3/linux64/bin/lwhostid
lwhostid = COMPOSITE=85B967ABE7D5
```

2. Execute the `lmhostid` command:

```
# /opt/lw-flexlm/v11.14.1.3/linux64/bin/lmhostid
lmhostid - Copyright (c) 1989-2017 Flexera Software LLC. All Rights Reserved.
The FlexNet host ID of this machine is "02420adc0004"
```

3. Execute the `hostname` utility to get the host name, for example:

```
# hostname
licserver
```

The hostname is `licserver` in this example.

NOTE: On some hosts, the `lmhostid` and `lwhostid` utilities may provide seven strings of hexadecimal digits. Use the first one as the `lmhostid` or `lwhostid` of the host.

CAUTION! Some Red Hat Enterprise Linux® 7 x86 (32/64 bit) systems do not have `eth*` network interfaces which are needed for FlexNet to work. Please ask your system administrator to add such an interface. Or you can do it yourself using the following steps.

1. Let's assume that the ethernet interface is named *`enp0s3`*. Rename the `ifcfg-enp0s3` file in `/etc/sysconfig/network-scripts` to `ifcfg-eth0`.
2. Edit the `/etc/sysconfig/network-scripts/ifcfg-eth0` and make the following changes:
   a. Replace *`enp0s3`* by `eth0`.
   b. Add the MAC address of the ethernet interface using the `HWADDR=`*`00:11:22:33:44:55`* tag.
3. Reboot the system and re-run `lwhostid` and `lmhostid` commands.

Setting up the License Files

After authorization, your licensing information file will be sent to you. An example of the file is below:

```
SERVER hostname COMPOSITE=lwhostid
    VENDOR lynxrtsd
    FEATURE LS52_TOOLS_linux_x86_64 lynxrtsd 1.0 27-oct-2012 1 \
        VENDOR_STRING=Lynx HOSTID=COMPOSITE=86d089b60c00 \
        NOTICE=common SIGN="0CBB EE99 95CC C7C7 0303 \
        3232 0000 7474 5757 5757 C888 4444 BCDE DBAC 0014 0444 BBBB \
        4444 3333 2222 1111 0000 9999 8888 7777 6666 5555 4444 3333 \
        D2FC"
```

This is the license information file that will be used on the host where the license server is to be run. The file can be placed in an arbitrary location. Set the `LM_LICENSE_FILE` environment variable to point the license file's location, then start the license manager daemon:

```
$ export LM_LICENSE_FILE=license_file_path
$ /opt/lw-flexlm/v11.14.1.3/linux64/bin/lmgrd
```

Multi-User Environment and Multiple Projects

Prior to performing any further LynxSecure tasks, the user should clone the LynxSecure environment into a local project directory.

CAUTION! Using the environment from `/opt/lynxsecure` directly, as it was recommended in previous releases, is no longer supported.

To set up a cloned environment, run the `CREATE_PROJECT` script from the LynxSecure installation directory:

```
$ /opt/lynxsecure/release/x86_64/CREATE_PROJECT
```

where the *`release`* is `6.3.0-rev16326` for the licensed version or `6.3.0-rev16326-eval` for the evaluation version.

The script will prompt for the location of the cloned copy to be created, and the target architectures which should be enabled. The location can be supplied using the `-d` option to this script, e.g.:

```
$ /opt/lynxsecure/6.3.0-rev16326/x86_64/CREATE_PROJECT -d new-project-location
```

The target architecture(s) can be specified using the `-a` option to this script, e.g.:

```
$ /opt/lynxsecure/6.3.0-rev16326/x86_64/CREATE_PROJECT -a x86_64 -a aarch64
```

After the script completes, the specified directory will be usable as a separate LynxSecure environment. To use that environment, source the `SETUP.bash` script from that newly created directory, e.g.:

```
$ source $HOME/lynxsecure-6.3.0-rev16326-x86_64/SETUP.bash
```

In the CDK version, after sourcing `SETUP.bash`, the license session manager should be started. As described above, the `LM_LICENSE_FILE` environment variable must be set up to point to the location of the license file. The license manager for a specific architecture can be started using the `{arch}-lwsmgr` command.

For the x86 platform:

```
$ x86_64-lwsmgr
```

For the Arm® platform:

```
$ aarch64-lwsmgr
```

Only one instance of `lwsmgr` per user needs to be started.

NOTE: To support workflows created for previous versions of the product, `lwsmgr` (without a specified architecture) is included as an alias for `x86_64-lwsmgr`.

Development Host Services Configuration

This section describes a basic LynxSecure development host configuration that enables the users to boot the Hardware Discovery Linux using the PXE boot standard and PXELINUX software. Later, the same boot mechanism can be used to boot the LynxSecure runtime images (SRPs).

PXE boot is a function built into many motherboards and network interface cards that enables a computer to boot an image received over the network. PXE boot requires two network services for correct operation: the Dynamic Host Configuration Protocol (DHCP) and the Trivial File Transfer Protocol (TFTP). Further, the Network File System (NFS) is an optional dependency for Hardware Discovery Linux.

Prerequisites and Assumptions

This section assumes the following:

1. CentOS 7.3 or later is running on the development host
2. LynxSecure has been installed on the development host.

3. The development host has two Ethernet interfaces, named `eth0` and `eth1`.
   a. `eth0` is assumed to connect to a local LAN or shared network
   b. `eth1` is connected directly to the LynxSecure target
4. The LynxSecure target supports PXE boot, and PXE boot has been enabled. Enabling PXE boot is generally accomplished via BIOS settings, but the exact procedure depends on the target. Consult user BIOS and network card documentation for details.

Network Configuration

The following steps will configure the network interface `eth1` with a static IP address and ensure that it is enabled by default.

1. To begin, log into the graphical desktop of the development host
2. Select **Applications → System Tools → Settings → Network**
3. Select **eth1** and click **the gear icon**; the **eth1** configuration window will now appear.
4. On the identity page, ensure that the checkbox **Connect Automatically** is checked
5. Select the ipv4 page. Ensure that the **IPv4 Toggle** is enabled. From the **Addresses** drop down select **Manual** and enter 10.20.31.1 as the IP address, 255.255.255.0 as the netmask, and 0.0.0.0 the gateway. At the bottom of the IPv4 page, check the box that says "Use this connection only for resources on its network".

NOTE: The values chosen for the IP Address and Netmask fields are examples. You may need to adjust these values to prevent conflicts with addresses ranges already in use on user network.

6. Click **Apply** and close the configuration window.
7. After applying the configuration to `eth1`, the network settings can be verified by running `ip addr`:

```
$ ip addr show dev eth1
6: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:25:80:7c brd ff:ff:ff:ff:ff:ff
    inet 10.20.31.1/24 brd 10.20.31.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::b00d:4bf6:bb44:11ce/64 scope link
       valid_lft forever preferred_lft forever
```

Enabling TFTP

Installing LynxSecure ensures that the TFTP software is already present on the LynxSecure host. However, this service is not enabled by default. To enable TFTP, run the following commands as root:

```
# systemctl enable tftp
# systemctl start tftp
```

After executing these commands, the tftp server will be running, and will be started after each boot. Files placed in `/var/lib/tftpboot` will be accessible via TFTP.

By default, CentOS configures `firewalld` to block TFTP traffic. To allow TFTP traffic to pass the firewall, run the following commands as root:

```
# firewall-cmd --zone=public --add-service=tftp --permanent
# firewall-cmd --reload
```

Configuring and Running the DHCP Server

Similar to TFTP, installation of LynxSecure ensures that the DHCP server is already installed. However, before enabling the DHCP server some configuration is required.

NOTE: Running multiple DHCP servers on a single network will cause DHCP to behave erratically. The configuration steps below ensure that the DHCP server is listening on a particular network. Be sure not to introduce subnet declarations that cause `dhcpd` to respond on an unintended interface.

First we will need a valid `/etc/dhcp/dhcpd.conf` file - an example file that can be used as a starting point can be found in `/opt/lynxsecure/6.3.0-rev16326/x86_64/examples/host/dhcpd.conf`. For convenience, we list its contents below:

```
# Notes:
# After placing this file in /etc/dhcp/dhcpd.conf, the DHCP server can
# be installed by running the following commands:
#
# systemctl enable dhcpd
# systemctl start dhcpd
#
# Note that this configuration will listen on any interface that has a matching
# subnet declaration. Be careful to insure that no subnet declarations are
# created for networks that already have a DHCP server elsewhere.

option arch code 93 = unsigned integer 16; # RFC4578

subnet 10.20.31.0 netmask 255.255.255.0 {
  option routers 10.20.31.254;

  pool {
    option domain-name-servers lynuxsecure-targets.example.com;
    max-lease-time 86400; # 24 hours
    range 10.20.31.2 10.20.31.253;
    allow unknown-clients;
    if option arch = 00:07 {
      filename "grubx64.efi";
    } else {
      filename "pxelinux.0";
    }
    server-name "10.20.31.1";
    next-server 10.20.31.1;
  }
}
```

Please update the subnet declaration to match the IP address that you assigned to eth1 above.

With `dhcpd` configured, it can be enabled by running:

```
$ sudo systemctl enable dhcpd
$ sudo systemctl start dhcpd
```

PXELINUX Configuration

PXELINUX runs on the target after the PXE boot loader is built into the target system. The PXELINUX configuration determines which software is loaded next, such as the Autoconfig Tool or a Separation Kernel Runtime Package.

PXELINUX configuration is stored in the `/var/lib/tftpboot/pxelinux.cfg` directory. Each file in that directory has a name that corresponds to either the MAC or IP address of the network interface the target machine uses to boot. For MAC-based names user must prepend `01-` before the dash-separated MAC address in the file name. For IP addresses based names, the IP address must be written as a 32-bit hexadecimal number. If PXELINUX cannot find a MAC-specific or IP-specific configuration for the target, it falls back to using the file named `default`.

If no default PXELINUX configuration is present, installing the LynxSecure SDK will create one. The default PXELINUX configuration can always be reinstalled manually by running the commands below.

```
$ source SETUP.bash
$ mkdir -p /var/lib/tftpboot/pxelinux.cfg
$ sudo cp /opt/lynxsecure/6.3.0-rev16326/x86_64_misc/pxelinux.cfg.default \
        /var/lib/tftpboot/pxelinux.cfg/default
```

If the provided example PXELINUX configuration is used, PXELINUX will load Hardware Discovery Linux by default. Once the target boots, the user can log in as root without a password. After logging in, it is possible to run the `ip addr` command to determine the target's IP address for use with the Autoconfig tool on the LynxSecure host.

For targets that require booting in UEFI mode, the default LynxSecure PXE configuration described above includes support via GRUB. The default configuration for UEFI booting can be found under `/var/lib/tftpboot/grub.cfg`.

NFS Server Configuration

LynxSecure Hardware Discovery Linux supports loading and storing data via NFS. During startup Hardware Discovery Linux attempts to mount an NFS share as defined by the `cdknfspath` boot parameter. The default value of the parameter is `10.20.31.1:/home/nfs`. If a different NFS share is desired, then it can be specified by modifying the PXE configuration file. For example `cdknfspath=192.168.0.1:/export/nfs` would mount /export/nfs instead of /home/nfs.

As with other needed services, NFS is installed as a dependency of LynxSecure, but is not enabled by default. To enable nfs:

1. First, create or edit `/etc/exports` to include the following:

   ```
   /home/nfs *(rw,sync,no_root_squash,no_subtree_check)
   ```

2. Next, create the folder described:

   ```
   # mkdir -p /home/nfs
   # chmod -R 755 /home/nfs
   # chown nfsnobody:nfsnobody /home/nfs
   ```

3. Setup the firewall:

   ```
   # firewall-cmd --permanent --zone=public --add-service=nfs
   # firewall-cmd --permanent --zone=public --add-service=mountd
   # firewall-cmd --permanent --zone=public --add-service=rpc-bind
   # firewall-cmd --reload
   ```

4. Enable and start the needed services:

   ```
   # systemctl enable rpcbind
   # systemctl enable nfs-server
   # systemctl enable nfs-lock
   # systemctl enable nfs-idmap
   # systemctl start rpcbind
   # systemctl start nfs-server
   # systemctl start nfs-lock
   # systemctl start nfs-idmap
   ```

Booting the Hardware Discovery Linux from a CD/DVD-ROM or USB Storage (only x86 platform)

NOTE: The information in this section applies only if the target platform is x86.

Before booting from a CD/DVD-ROM or USB storage, device support should be enabled in the target machine BIOS.

To create bootable CD/DVD-ROM or USB storage, the image `$ENV_PREFIX/build/autoconfig.iso` shall be written to the media.

Any standard CD/DVD burning tool can be used to write `autoconfig.iso` to the media. Use the following command to create bootable USB storage:

```
$ dd if=$ENV_PREFIX/autoconfig/autoconfig.iso \
        of=/dev/sdX
```

CHAPTER 3 *Setting up the Target System*

LynxSecure® Development Workflow Step 2

Target systems running LynxSecure must be pre-configured to enable required features in the system firmware. For example, this step may include enabling VT-d support on x86 platforms.

Setting up x86 Targets

Enter the system BIOS configuration menu and make the following settings:

- Enable Intel® Virtualization Technology (VT-x)
- Enable Intel Virtualization Technology for Directed I/O (VT-d)
- Enable PXE if booting from the network
- Disable Intel TXT (it is not supported by LynxSecure)
- Disable "Memory Mapped IO above 4GB" or similar options

Creating Partitions on the Storage Device

Usually, partitioning the storage device is required when Block Device Emulation is used with multiple subjects. Please make sure that the target system's storage devices are partitioned properly before collecting the target hardware information using the Autoconfig Tool as described in the following chapters.

The user may use any disk partitioning utility included in the Hardware Discovery Linux® to create the partition table. Some of them are:

- MBR partitioning tools: `fdisk` (interactive), `cfdisk` (ncurses-based), `sfdisk`.
- GPT partitioning tools: `gdisk` (interactive), `cgdisk` (ncurses-based), `sgdisk`.

NOTE: We recommend creating a GPT partition table rather than MBR: the Autoconfig Tool, discussed in Chapter 4, "*Introduction to the Autoconfig Tool*" (page 15), captures the information about the GPT labels (partition aliases), which can be used from the Autoconfig Tool command line when creating configurations with storage device emulation.

Below is an example of creating a partition layout using the `sgdisk` on a `/dev/sda` device:

- Create a clean GUID Partition Table (GPT):

```
sgdisk -og /dev/sda
```

If the disk's partition structures are damaged, it may be necessary to reset them first using the `sgdisk -Z` command.

- Add a new 10G partition #1:

```
sgdisk --new 1::+10G -c 1:gpt_label_10G /dev/sda
```

- Add a new 55G partition #2:

```
sgdisk --new 2::+55G -c 2:gpt_label_55G /dev/sda
```

- And so forth...
- At the end, the user may want to display the partition layout:

```
sgdisk -p /dev/sda
```

The Autoconfig Tool, discussed in Chapter 4, "*Introduction to the Autoconfig Tool*" (page 15), assigns a special meaning to several GPT labels:

Table 3-1: LynxSecure System Partitions

GPT Label	Description
`vds0root`	GPT label for the HDD-based VDS root filesystem.
`vds0`	GPT label for the external storage which is mounted at the VDS startup (so called "VDS persistent storage").
`vds0swap`	GPT label for the HDD-based swap space.

Setting up Xilinx Zynq Ultrascale+ MPSoC Targets

- Set the switches for SD card boot as described, for example, at the bottom of this page [https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/45940846/Zynq+UltraScale+MPSoC+Base+TRD+2018.3]. If you need the USB controller to operate in host mode rather than endpoint mode, the same page has instructions for setting the jumpers accordingly as well.
- Install Xilinx® Software Release 2018.3 found at Xilinx Wiki [http://www.wiki.xilinx.com/Zynq+Releases] to an SD card using the procedure described in that document. This should enable the U-Boot bootloader on the board.
- Monitors that are known to work with the display port on the Xilinx Zynq® Ultrascale+™ MPSoC board: https://www.xilinx.com/support/answers/68671.html
- `zynqmp-fv` is provided as a base buildroot configuration for creating a FV buildroot subject.

NOTE: Using a Xilinx Software Release other than the one mentioned above is not supported and may result in issues.

Setting up NXP S32V234 MPSoC Targets

- The NXP® S32V234™ MPSoC target has been verified with the U-Boot and Linux released in the board support package (BSP) version 0.6 from the board manufacturer. Using other versions may result in issues.
- The default U-Boot configuration provided by the NXP at http://git.freescale.com/git/cgit.cgi/auto/u-boot.git/ cannot boot LynxSecure. It is configured to start the secondary CPUs in EL1 exception level, which is not

sufficient to run a hypervisor. It also limits the size of the bootable image to 8Mb, which is not sufficient for many configurations. Furthermore, it does not support the Flattened Image Tree (FIT) format for bootable images.

LynxSecure provides a modified version of U-Boot that addresses these issues. A prebuilt version of U-Boot is available as `aarch64/libexec/u-boot-s32v234evb.s32`; to install on an SD card, execute the following command, replacing */dev/sdX* with the block device name for the SD card on the development host:

```
$ dd if=aarch64/libexec/u-boot-s32v234evb.s32 of=/dev/sdX bs=512 seek=8 conv=fsync
```

Refer to the board's documentation for further details on setting up the SD card.

CHAPTER 4 *Introduction to the Autoconfig Tool*

The Autoconfig Tool is a command line tool that simplifies the creation of bootable LynxSecure® runtime images (SRPs) for the users. It can also be used to convert a configuration expressed in simple syntax on the command line to the corresponding LynxSecure XML configuration file (HCV). Another function of the tool is collecting the hardware information for target systems and storing it in an internal database, which allows for easy target and device referencing on the command line.

The tool provides a set of configuration options for creating a required LynxSecure configuration, such as: how many guests are required, which hardware is assigned to which guest, how much memory should be used for a particular guest, which virtual devices should be emulated for a particular guest, etc.

Working with Target System Information

Collecting Target Information on x86 Platform

LynxSecure Development Workflow Step 3

Once the target is pre-configured, users must then run a probe tool from the development host. This tool captures the required target hardware and configuration profile needed to start the configuration process.

For target systems based on the x86 platform, a special Linux® bootable image is used for collecting target system information, which is called the Hardware Discovery Linux. This image should be booted on the target system (e.g. using either the PXE server, CD/DVD ROM or the USB storage) prior to collecting system information. Then, the Autoconfig Tool should be launched on the development host with appropriate arguments to transfer the target system information to the development host.

The target system information (such as the CPU topology, chipset information, peripheral components information, etc.) is stored on the development host in `$ENV_PREFIX/build/db/`*`TARGET-NAME`* directory and may be used for creating LynxSecure runtime packages (the user still can create an HCV by hand). Note, that if the hardware setup changes on the target (a new peripheral component is installed, BIOS settings are changed, partition tables are altered, and so on), it is necessary to recollect the hardware information.

Synopsis

```
autoconfig collect TARGET-NAME IP-ADDRESS[:PORT]|HOSTNAME[:PORT]|FILENAME \
  [--device-list=[assignable|disabled][,...]] [--extra-logs] [-g]
```

Description

This command creates a hardware database for the specified target.

For x86 targets, the user must boot the Hardware Discovery Linux on the target system and use this command with the target's hostname or IP address to collect the hardware information. In case the target has no network connection, the hardware information may be still collected by running the Autoconfig Tool on the target directly (assuming that the Hardware Discovery Linux is booted on the target):

```
$ /opt/lynxsecure/6.3.0-rev16326/x86_64/CREATE_PROJECT -d DIR
$ cd DIR
$ . SETUP.bash
$ autoconfig collect TARGET-NAME 127.0.0.1
```

For x86 targets, `--extra-logs` option (enabled by default) also enforces transferring of the log files from the target system, such as `/proc` entries, `dmesg` and ACPI tables (both raw and decoded). The `$ENV_PREFIX/build/db/`*`TARGET-NAME`*`.tar.bz2` contains all information about the target system and may be requested by the Lynx Software Technologies, Inc. customer support team.

Collecting Target Information on Arm Platform

LynxSecure Development Workflow Step 4

Once the target is pre-configured, users must then retrieve the target information using a command line tool. This tool captures the required target hardware and configuration profile needed to start the configuration process.

For Arm® targets, the source of the hardware information is the DTB file (Devicetree blob) which is supplied with LynxSecure for each target board. Use the DTB filename instead of the hostname or IP address. The Devicetree blobs are located in `$ENV_PREFIX/devicetree/aarch64/<boardname>/<boardname>.dtb`. The following command can be used to create a hardware database for the Xilinx® Zynq® Ultrascale+™ MPSoC target:

```
$ autoconfig collect TARGET-NAME $ENV_PREFIX/aarch64/libexec/zynqmp-zcu102.dtb
```

Table 4-1: Platform Devices verified on Xilinx Zynq Ultrascale+ MPSoC

Platform Device	Description	FV subject direct assignment	Special Note
AHCI0	ceva,ahci-1v84	Verified	The SATA/AHCI device may be configured as a Secure device by the board manufacturer on the ZynqMP board. If that is the case, the device won't be functional in regular (not identity-mapped) subjects, as its DMA transactions would bypass the SMMU. This problem can be fixed by re-configuring the device as Non-secure before LynxSecure boots, or using `identitymem` for FV direct assignment. To switch into Non-Secure mode, the register `slcr_sata` (0xFD690020) must be set to the value 0xF. Note that this register can only be programmed in Secure mode, EL3. The board manufacturer recommends that this register is set before the First Stage Boot Loader passes control to the next stage bootloader (U-Boot). This issue cannot be fixed in LynxSecure, which runs at privilege level EL2.
DMA0	xlnx,dpdma	Verified (as dependency of ZYNQMP-DISPLAY0)	

Platform Device	Description	FV subject direct assignment	Special Note
ZYNQMP-DISPLAY0	xlnx,zynqmp-dpsub-1.7	Verified	• LynxSecure unconditionally enables SMMU translation for devices assigned to subjects on the Ultrascale+. Xilinx has indicated that there may be latency-related issues with enabling SMMU translation for the DisplayPort device. While Lynx Software Technologies, Inc. has not yet encountered such issues, we recommend assigning the DisplayPort device to an identity-mapped subject to simplify the transition to an SMMU-bypassed configuration for this device if and when LynxSecure supports such a SMMU bypass to mitigate this issue. • Needs the following dependent devices: `DMA0`
GPU0	arm,mali-400	Verified	• Needs `identitymem` for FV direct assignment as GPU DMA controller always does secure [https://www.xilinx.com/support/answers/69052.html] transactions which bypasses the non-secure SMMU translations configured by the SKH.
NET3	cdns,zynqmp-gem	Verified	
PCIE0	xlnx,nwl-pcie-2.11	Verified with following Intel® NIC cards: • PCI ID 8086:10d3 • PCI ID 8086:107d	Assigning this device to a FV subject will also assign every PCIe device connected beneath it to the same FV subject.
PHY0	xlnx,zynqmp-psgtr		A number of physical devices (AHCI, PCIE, ZYNQMP-DISPLAY, USB) use the PS-GTR PHY device (PHY0) and are dependent on it in the board's Devicetree. Those physical devices and PHY0 should be assigned to the same subject. If not, driver software for those devices may need to be modified, as it could be attempting to program the PHY.
SERIAL[0,1]	cdns,uart-r1p12	Verified	
USB0	xlnx,zynqmp-dwc3	Verified	

The following command can be used to create a hardware database for the NXP® S32V234™ MPSoC target:

```
$ autoconfig collect TARGET-NAME $ENV_PREFIX/aarch64/libexec/s32v234-evb28899.dtb
```

Table 4-2: Platform Devices verified on NXP S32V234 MPSoC

Platform Device	Description	FV subject direct assignment	Special Note
I2C[0-2]	fsl,s32v234-i2c	Verified	• Needs `identitymem` for a FV direct assignment. • Needs the following dependent device: `DMA1`
NET0	fsl,s32v234-fec	Verified	

Platform Device	Description	FV subject direct assignment	Special Note
SERIAL[0-1]	fsl,s32-linflexuart	Verified	• Needs `identitymem` for a FV direct assignment if `DMA1` is also available. • Dependent on the following device (if available): `DMA1`
MMC0	fsl,s32v234-usdhc	Verified	• Needs `identitymem` for a FV direct assignment. • Needs the following dependent device: `DMA1`

CAUTION! The Devicetree blobs supplied with LynxSecure have been customized for LynxSecure; do not use the vanilla Devicetree blobs provided by the board manufacturer with the Autoconfig Tool command.

Customizing the Device Tree

Every manual Devicetree spec (.dts) file modification must be followed by `dtbtool` invocation. The tool inserts a special node called `__local_fixups__` which is not *necessarily* supported by the stock version of DTC (Devicetree Compiler) supplied with the pristine Linux sources for the *<boardname>*. The `__local_fixups__` is used to record the device dependencies which, in turn, are used for resolving device assignments at the SRP creation time.

CAUTION! Not invoking `dtbtool` after manual modification of .dts files may lead to creating an SRP which will fail to function properly or even fail to boot.

Displaying the Target Hardware Information

Synopsis

```
autoconfig show target-name[--device-list=[assignable|disabled][,...]] [-g]
```

Description

This command displays the information collected by the Autoconfig Tool from the target *target-name*. There are three sections:

- General System Information (such as number of CPUs, amount of RAM available, support for VT-x/VT-d technologies)
- Disk Information (such as all available partitions on all storage devices attached to the system, PCI addresses of controllers to which the disks are attached)
- Peripheral Device Information (such as PCI, USB and legacy devices found on the target system)

An example of the output from this command is provided below:

```
$ autoconfig show x8dah
Arch              x86_64
Product Name      X8DAH
Manufacturer      Supermicro
Version           1234567890
Serial Number     1234567890
System RAM        0x2ff700000/11.99GiB
System RAM (low)  0xbf700000/2.99GiB
Virtual VGA       10 subjects
IOMMU             Yes
CPU Model         Intel(R) Xeon(R) E5520 2.27GHz
CPU Cores         8
CPU Threads       16
HT                Yes
```

```
Memory Map
==========
0000000000000000-000000000009ffff 00000000000a0000 LOWMEM
00000000000a0000-00000000000bffff 0000000000020000 PCI Bus 0000:00
00000000000c0000-00000000000c7fff 0000000000008000 Video ROM
00000000000d0000-00000000000dffff 0000000000010000 PCI Bus 0000:00
00000000000e4000-00000000000e7fff 0000000000004000 Reserved memory Region Reporting Structure 0000:00:1d.0,0000:00:1d.1,0000:00:1d
00000000000e8000-00000000000fffff 0000000000018000 reserved
0000000000100000-00000000bf75ffff 00000000bf660000 System RAM
00000000bf760000-00000000bf76ffff 0000000000010000 reserved
00000000bf770000-00000000bf77dfff 000000000000e000 ACPI Tables
00000000bf77e000-00000000bf7cffff 0000000000052000 ACPI Non-volatile Storage
00000000bf7d0000-00000000bf7ebfff 000000000001c000 reserved
00000000bf7ec000-00000000bf7fffff 0000000000014000 Reserved memory Region Reporting Structure 0000:00:1d.0,0000:00:1d.1,0000:00:1d
00000000bf800000-00000000bfffffff 0000000000800000 reserved
00000000cd000000-00000000cddfffff 0000000000e00000 PCI Bus 0000:80
00000000cde00000-00000000cdefffff 0000000000100000 PCI Bus 0000:83
00000000cdf00000-00000000cdffffff 0000000000100000 PCI Bus 0000:85
00000000ce000000-00000000dfffffff 0000000012000000 PCI Bus 0000:03
00000000e0000000-00000000efffffff 0000000010000000 PCI MMCONFIG 0000 [bus 00-ff]
00000000f0000000-00000000f01fffff 0000000000200000 PCI Bus 0000:06
00000000f0200000-00000000f6ffffff 0000000006e00000 PCI Bus 0000:00
00000000f7000000-00000000f7ffffff 0000000001000000 PCI Bus 0000:07
00000000f8000000-00000000f8dfffff 0000000000e00000 PCI Bus 0000:00
00000000f8e00000-00000000f8ffffff 0000000000200000 PCI Bus 0000:01
00000000f9000000-00000000faefffff 0000000001f00000 PCI Bus 0000:03
00000000faf00000-00000000faffffff 0000000000100000 PCI Bus 0000:06
00000000fb000000-00000000fbefffff 0000000000f00000 PCI Bus 0000:07
00000000fbf00000-00000000fbffffff 0000000000100000 PCI Bus 0000:00
00000000fec00000-00000000fec00fff 0000000000001000 IOAPIC 0
00000000fed00000-00000000fed00fff 0000000000001000 HPET 0
00000000fed1c000-00000000fed1ffff 0000000000004000 pnp 00:00
00000000fed20000-00000000fed3ffff 0000000000020000 pnp 00:06
00000000fed40000-00000000fed44fff 0000000000005000 PCI Bus 0000:00
00000000fed45000-00000000fed8ffff 000000000004b000 pnp 00:06
00000000fee00000-00000000fee00fff 0000000000001000 Local APIC
00000000ffc00000-00000000ffffffff 0000000000400000 reserved
0000000100000000-000000033fffffff 0000000240000000 System RAM

Block Devices
===========
Controller Type Size       Persistent ID                          GPT label            Device ID
========== ==== ====       =============                          =========            =======
AHCI0      hd   465.76GiB  hd-0000:00:1f.2-0.0.0.0-0                                   wwn-0x50014ee602493070
AHCI0      hd   1023.02MiB hd-0000:00:1f.2-0.0.0.0-1              lsk_boot             wwn-0x50014ee602493070-part1
AHCI0      hd   25.00GiB   hd-0000:00:1f.2-0.0.0.0-10             win7.64              wwn-0x50014ee602493070-part10
AHCI0      hd   25.00GiB   hd-0000:00:1f.2-0.0.0.0-11             win7.32              wwn-0x50014ee602493070-part11
AHCI0      hd   20.00GiB   hd-0000:00:1f.2-0.0.0.0-12             winxp.32             wwn-0x50014ee602493070-part12
AHCI0      hd   25.00GiB   hd-0000:00:1f.2-0.0.0.0-13             netbsd.64            wwn-0x50014ee602493070-part13
AHCI0      hd   25.00GiB   hd-0000:00:1f.2-0.0.0.0-14             netbsd.32            wwn-0x50014ee602493070-part14
AHCI0      hd   20.00GiB   hd-0000:00:1f.2-0.0.0.0-15             winxp.64             wwn-0x50014ee602493070-part15
AHCI0      hd   13.97GiB   hd-0000:00:1f.2-0.0.0.0-16                                  wwn-0x50014ee602493070-part16
AHCI0      hd   4.00MiB    hd-0000:00:1f.2-0.0.0.0-17                                  wwn-0x50014ee602493070-part17
AHCI0      hd   8.00GiB    hd-0000:00:1f.2-0.0.0.0-2              vds0swap             wwn-0x50014ee602493070-part2
AHCI0      hd   1023.02MiB hd-0000:00:1f.2-0.0.0.0-3              vds0                 wwn-0x50014ee602493070-part3
AHCI0      hd   25.00GiB   hd-0000:00:1f.2-0.0.0.0-4              freebsd10.64         wwn-0x50014ee602493070-part4
AHCI0      hd   25.00GiB   hd-0000:00:1f.2-0.0.0.0-5              freebsd10.32         wwn-0x50014ee602493070-part5
AHCI0      hd   25.00GiB   hd-0000:00:1f.2-0.0.0.0-6              ubuntu16.64          wwn-0x50014ee602493070-part6
AHCI0      hd   25.00GiB   hd-0000:00:1f.2-0.0.0.0-7              ubuntu16.32          wwn-0x50014ee602493070-part7
AHCI0      hd   40.00GiB   hd-0000:00:1f.2-0.0.0.0-8              win10                wwn-0x50014ee602493070-part8
AHCI0      hd   30.00GiB   hd-0000:00:1f.2-0.0.0.0-9              win8                 wwn-0x50014ee602493070-part9
AHCI0      cd   N/A        cd-0000:00:1f.2-2.0.0.0-0                                   ata-Optiarc_DVD_RW_AD-7220S
RAID0      hd   51.24GiB   hd-0000:83:00.0-0.0.0.0-0                                   wwn-0x5e83a97f33714cdd
RAID0      hd   51.24GiB   hd-0000:83:00.0-0.0.0.0-1                                   wwn-0x5e83a97f33714cdd-part1
RAID0      hd   1.57MiB    hd-0000:83:00.0-0.0.0.0-2                                   wwn-0x5e83a97f33714cdd-part2
RAID0      hd   51.24GiB   hd-0000:83:00.0-1.0.0.0-0                                   wwn-0x5e83a97ff0237d6c
RAID0      hd   51.24GiB   hd-0000:83:00.0-1.0.0.0-1                                   wwn-0x5e83a97ff0237d6c-part1
RAID0      hd   51.24GiB   hd-0000:83:00.0-2.0.0.0-0                                   wwn-0x5e83a97f3129ae4d
RAID0      hd   51.24GiB   hd-0000:83:00.0-2.0.0.0-1                                   wwn-0x5e83a97f3129ae4d-part1
RAID0      hd   51.24GiB   hd-0000:83:00.0-3.0.0.0-0                                   wwn-0x5e83a97f3c3759ce
RAID0      hd   51.24GiB   hd-0000:83:00.0-3.0.0.0-1                                   wwn-0x5e83a97f3c3759ce-part1
RAID1      hd   51.24GiB   hd-0000:85:00.0-0.0.0.0-0                                   wwn-0x5e83a97f27be0f17
RAID1      hd   51.24GiB   hd-0000:85:00.0-0.0.0.0-1                                   wwn-0x5e83a97f27be0f17-part1
RAID1      hd   51.24GiB   hd-0000:85:00.0-1.0.0.0-0                                   wwn-0x5e83a97f7688ec4b
RAID1      hd   51.24GiB   hd-0000:85:00.0-1.0.0.0-1                                   wwn-0x5e83a97f7688ec4b-part1
RAID1      hd   51.24GiB   hd-0000:85:00.0-2.0.0.0-0                                   wwn-0x5e83a97fcfcd5896
RAID1      hd   51.24GiB   hd-0000:85:00.0-2.0.0.0-1                                   wwn-0x5e83a97fcfcd5896-part1
RAID1      hd   51.24GiB   hd-0000:85:00.0-3.0.0.0-0                                   wwn-0x5e83a97f54b73be7
RAID1      hd   51.24GiB   hd-0000:85:00.0-3.0.0.0-1                                   wwn-0x5e83a97f54b73be7-part1

USB Devices
===========
Controller Device Path                      Device ID HW Rev Serial Number              Class:Subclass:Procotol                  C
========== ===========                      ========= ====== =============              =======================                  =
USB0       usb-0000:00:1a.0-full            1d6b:0001 0409   0000:00:1a.0               09:00:00
USB1       usb-0000:00:1a.1-full            1d6b:0001 0409   0000:00:1a.1               09:00:00
USB2       usb-0000:00:1a.2-full            1d6b:0001 0409   0000:00:1a.2               09:00:00
USB2       usb-0000:00:1a.2-full-2          046b:ff10 0100   serial                     03:01:01:0001:0006,03:01:02:0001:0002    U
USB3       usb-0000:00:1a.7-high            1d6b:0002 0409   0000:00:1a.7               09:00:00
USB4       usb-0000:00:1d.0-full            1d6b:0001 0409   0000:00:1d.0               09:00:00
```

```
USB5       usb-0000:00:1d.1-full            1d6b:0001 0409   0000:00:1d.1               09:00:00
USB6       usb-0000:00:1d.2-full            1d6b:0001 0409   0000:00:1d.2               09:00:00
USB7       usb-0000:00:1d.7-high            1d6b:0002 0409   0000:00:1d.7               09:00:00

PCI Devices
===========
      Name             Device ID Address                           Comment Description
      ====             ========= =======                           ======= ===========
      PIC0             8086342e  0000:00:14.0                              PIC: Intel Corporation 7500/5520/5500/X58 I/O Hub Syste
      PIC1             80863422  0000:00:14.1                              PIC: Intel Corporation 7500/5520/5500/X58 I/O Hub GPIO
      PIC2             80863423  0000:00:14.2                              PIC: Intel Corporation 7500/5520/5500/X58 I/O Hub Contr
      PIC3             80863438  0000:00:14.3                              PIC: Intel Corporation 7500/5520/5500/X58 I/O Hub Throt
      PCIDEV0          80863430  0000:00:16.0                              System peripheral: Intel Corporation 5520/5500/X58 Chip
      PCIDEV1          80863431  0000:00:16.1                              System peripheral: Intel Corporation 5520/5500/X58 Chip
      PCIDEV2          80863432  0000:00:16.2                              System peripheral: Intel Corporation 5520/5500/X58 Chip
      PCIDEV3          80863433  0000:00:16.3                              System peripheral: Intel Corporation 5520/5500/X58 Chip
      PCIDEV4          80863429  0000:00:16.4                              System peripheral: Intel Corporation 5520/5500/X58 Chip
      PCIDEV5          8086342a  0000:00:16.5                              System peripheral: Intel Corporation 5520/5500/X58 Chip
      PCIDEV6          8086342b  0000:00:16.6                              System peripheral: Intel Corporation 5520/5500/X58 Chip
      PCIDEV7          8086342c  0000:00:16.7                              System peripheral: Intel Corporation 5520/5500/X58 Chip
      USB0             80863a37  0000:00:1a.0                              USB controller: Intel Corporation 82801JI (ICH10 Family
      USB1             80863a38  0000:00:1a.1                              USB controller: Intel Corporation 82801JI (ICH10 Family
      USB2             80863a39  0000:00:1a.2                              USB controller: Intel Corporation 82801JI (ICH10 Family
      USB3             80863a3c  0000:00:1a.7                              USB controller: Intel Corporation 82801JI (ICH10 Family
      AUDIO0           80863a3e  0000:00:1b.0                              Audio device: Intel Corporation 82801JI (ICH10 Family)
      USB4             80863a34  0000:00:1d.0                              USB controller: Intel Corporation 82801JI (ICH10 Family
      USB5             80863a35  0000:00:1d.1                              USB controller: Intel Corporation 82801JI (ICH10 Family
      USB6             80863a36  0000:00:1d.2                              USB controller: Intel Corporation 82801JI (ICH10 Family
      USB7             80863a3a  0000:00:1d.7                              USB controller: Intel Corporation 82801JI (ICH10 Family
      LPC0             80863a16  0000:00:1f.0                              ISA bridge: Intel Corporation 82801JIR (ICH10R) LPC Int
      AHCI0            80863a22  0000:00:1f.2                              SATA controller: Intel Corporation 82801JI (ICH10 Famil
      SMBUS0           80863a30  0000:00:1f.3                              SMBus: Intel Corporation 82801JI (ICH10 Family) SMBus C
      NET0             808610c9  0000:01:00.0            00:25:90:01:34:CA Ethernet controller: Intel Corporation 82576 Gigabit Ne
      NET1             808610c9  0000:01:00.1            00:25:90:01:34:CB Ethernet controller: Intel Corporation 82576 Gigabit Ne
      NET0#0           808610ca  0000:01:10.0                          VF0 Ethernet controller: Intel Corporation 82576 Gigabit Ne
      NET1#0           808610ca  0000:01:10.1                          VF0 Ethernet controller: Intel Corporation 82576 Gigabit Ne
      NET0#1           808610ca  0000:01:10.2                          VF1 Ethernet controller: Intel Corporation 82576 Gigabit Ne
      NET1#1           808610ca  0000:01:10.3                          VF1 Ethernet controller: Intel Corporation 82576 Gigabit Ne
      NET0#2           808610ca  0000:01:10.4                          VF2 Ethernet controller: Intel Corporation 82576 Gigabit Ne
      NET1#2           808610ca  0000:01:10.5                          VF2 Ethernet controller: Intel Corporation 82576 Gigabit Ne
      NET0#3           808610ca  0000:01:10.6                          VF3 Ethernet controller: Intel Corporation 82576 Gigabit Ne
      NET1#3           808610ca  0000:01:10.7                          VF3 Ethernet controller: Intel Corporation 82576 Gigabit Ne
      NET0#4           808610ca  0000:01:11.0                          VF4 Ethernet controller: Intel Corporation 82576 Gigabit Ne
      NET1#4           808610ca  0000:01:11.1                          VF4 Ethernet controller: Intel Corporation 82576 Gigabit Ne
      NET0#5           808610ca  0000:01:11.2                          VF5 Ethernet controller: Intel Corporation 82576 Gigabit Ne
      NET1#5           808610ca  0000:01:11.3                          VF5 Ethernet controller: Intel Corporation 82576 Gigabit Ne
      NET0#6           808610ca  0000:01:11.4                          VF6 Ethernet controller: Intel Corporation 82576 Gigabit Ne
      NET1#6           808610ca  0000:01:11.5                          VF6 Ethernet controller: Intel Corporation 82576 Gigabit Ne
      NET0#7           808610ca  0000:01:11.6                          VF7 Ethernet controller: Intel Corporation 82576 Gigabit Ne
      NET1#7           808610ca  0000:01:11.7                          VF7 Ethernet controller: Intel Corporation 82576 Gigabit Ne
      GRAPHICS0        10de0a78  0000:03:00.0                      Primary VGA compatible controller: NVIDIA Corporation GT218GL [
      AUDIO1           10de0be3  0000:03:00.1                              Audio device: NVIDIA Corporation High Definition Audio
      IDE0             197b2368  0000:06:00.0                              IDE interface: JMicron Technology Corp. JMB368 IDE cont
      GRAPHICS1        102b0532  0000:07:04.0                    Secondary VGA compatible controller: Matrox Electronics Systems L
      FIREWIRE0        104c8023  0000:07:05.0                              FireWire (IEEE 1394): Texas Instruments TSB43AB22A IEEE
      PIC4             8086342e  0000:80:14.0                              PIC: Intel Corporation 7500/5520/5500/X58 I/O Hub Syste
      PIC5             80863422  0000:80:14.1                              PIC: Intel Corporation 7500/5520/5500/X58 I/O Hub GPIO
      PIC6             80863423  0000:80:14.2                              PIC: Intel Corporation 7500/5520/5500/X58 I/O Hub Contr
      PIC7             80863438  0000:80:14.3                              PIC: Intel Corporation 7500/5520/5500/X58 I/O Hub Throt
      PCIDEV8          80863430  0000:80:16.0                              System peripheral: Intel Corporation 5520/5500/X58 Chip
      PCIDEV9          80863431  0000:80:16.1                              System peripheral: Intel Corporation 5520/5500/X58 Chip
      PCIDEV10         80863432  0000:80:16.2                              System peripheral: Intel Corporation 5520/5500/X58 Chip
      PCIDEV11         80863433  0000:80:16.3                              System peripheral: Intel Corporation 5520/5500/X58 Chip
      PCIDEV12         80863429  0000:80:16.4                              System peripheral: Intel Corporation 5520/5500/X58 Chip
      PCIDEV13         8086342a  0000:80:16.5                              System peripheral: Intel Corporation 5520/5500/X58 Chip
      PCIDEV14         8086342b  0000:80:16.6                              System peripheral: Intel Corporation 5520/5500/X58 Chip
      PCIDEV15         8086342c  0000:80:16.7                              System peripheral: Intel Corporation 5520/5500/X58 Chip
      RAID0            10953124  0000:83:00.0                              RAID bus controller: Silicon Image, Inc. SiI 3124 PCI-X
      RAID1            10953124  0000:85:00.0                              RAID bus controller: Silicon Image, Inc. SiI 3124 PCI-X

Platform Devices
================
      Name             Device ID Address                                    Description
      ====             ========= =======                                    ===========
      ACPI0            ffff0f01                                             Legacy ACPI
      CMOSRTC0         00000b00                                             CMOS realtime clock
      KEYBOARD0        0000030b                                             IBM Enhanced (101/102-key, PS/2 mouse support)
      LEGACYIDE0       ffff0303                                             Primary Legacy IDE Controller
      LEGACYIDE1       ffff0303                                             Secondary Legacy IDE Controller
      MOUSE0           00000f13                                             PS/2 port for PS/2-style mouse
      SERIAL0          00000501                                             16550A-compatible COM port at 0x3f8
      SERIAL1          00000501                                             16550A-compatible COM port at 0x2f8
      SERIAL2          00000501                                             16550A-compatible COM port at 0x3e8
      VGA0             00000900                                             VGA compatible device

I2C Devices
===========
      Name             Device ID Address                                    Description
      ====             ========= =======                                    ===========
```

Please refer to section “Commands” (page 125) for more information on various requests to the target system database.

NOTE: Adding a `-g` option will display more information, such as: I/O port ranges, I/O device memory ranges and IRQs.

Some devices may be assigned by dependency. That is, if a device assigned to a subject depends on another device, the Autoconfig Tool may automatically assign the other device to the same subject as well.

Some devices on Arm boards may have a "disabled" status in the DTB. Such devices are not visible to the user and disallowed for assignments. To display just the disabled devices, the `--device-list=disabled` shall be used. To display all assignable devices, the `--device-list=assignable,disabled` shall be used. To force a disabled device assignment, `--force` shall be used.

Working with Target System Information on the Development Host

Displaying the List of All Cached Targets

Synopsis

```
autoconfig targets
```

Description

This command shows the names of all targets that have their hardware information stored in the database.

Deleting Target Hardware Information from a Database

Synopsis

```
autoconfig delete target-name
```

Description

This command permanently deletes the information about the *`target-name`* target from the Autoconfig Tool's database.

Renaming a Target in the Database

Synopsis

```
autoconfig rename old-name new-name
```

Description

This command renames the target named *`old-name`* in the database to *`new-name`*.

Importing Databases

Synopsis

```
autoconfig import DIRECTORY [target1 target2]
```

Description

This command import target information from the specified *DIRECTORY*.

Exporting Databases

Synopsis

```
autoconfig export DIRECTORY [target1 target2]
```

Description

This command exports target information to the specified *DIRECTORY*.

Creating an SRP

LynxSecure Development Workflow Step 5

Next, a system configuration is generated using command line tools. This configuration is stored as XML and fully defines the run time configuration of the system while executing LynxSecure. This XML configuration is called the Human-readable Configuration Vector, and is often referred to as the HCV.

Synopsis

```
autoconfig mksrp target-name \
 --xmlname=xml-name,... \
 --subject-name=subjectsspec,... \
        --output=directory
```

Description

- The `mksrp` command creates `.xml` and `.srp` files for the system named as *`target-name`*. By default, the XML and SRP files named as *`target-name`*`.xml` and *`target-name`*`.srp` would be created in `$ENV_PREFIX/build/`*`target-name`* directory. The name of the XML and SRP files can be changed using the `--xmlname=`*`xml-name`* option.
- The option `--output=`*`directory`* can be used to copy the complete SRP image to the specified directory. This argument is commonly used to copy the SRP to the TFTP directory which is `/var/lib/tftpboot` by default.
- The *`subjectsspec`* is a list of subject specifications separated by commas. Please refer to Chapter 7, "*Subjects*" (page 45) for more details.

Creating the XML File (HCV) Without Creating an SRP

Synopsis

```
autoconfig mkhcv target-name \
--xmlname=xml-name,... \
--subject-name=subjectsspec,... \
```

Description

Sometimes it may be desired to create the XML configuration (HCV) without building an SRP; for example, to apply some manual changes to the configuration before creating the image. With this command, the Autoconfig Tool generates a `target-name.xml` file in the `$ENV_PREFIX/build/target-name` directory. The XML file name can be changed using the `--xmlname` option.

System-wide Options When Creating an SRP or HCV

Synopsis

```
autoconfig mksrp target-name \
  --arch=[aarch64|x86_64] \
  --auditsize=NUMBER \
  --diagoutput=[vga|usb|serial=[[SERIALx|HEX][:BAUDRATE[:MAX_BAUDRATE]]],...],... \
  --jsonpath=FILENAME \
  --dtbpath=FILENAME \
  --skhheap=[ADDRESS,]NUMBER[b|M|G] \
  --modules=MODULES \
  --srploadaddr=HPA \
  --ht \
  --iommu \
  --runtimediag=[none|fatalonly|all] \
  --smidisable \
  --tickspersec=NUMBER \
  --rifargs=STRING \
  -g[g..] \
  ...
```

Global Options

`--arch=[aarch64|x86_64]`

Select the architecture for the SRP. This option is usually unnecessary, because the target database contains the architecture of the target.

`--auditsize=`*`NUMBER`*

Set the maximum number of entries in the LynxSecure audit buffer.

`--diagoutput=[[vga|usb|serial=[SERIAL`*`x`***`|`***`HEX`***`][,`***`BAUDRATE`***`][,[`***`MAX_BAUDRATE`***`]]],...],...`**

Enable/disable LynxSecure Separation Kernel Hypervisor diagnostic output (comma-separated list). Use an empty value to disable the diagnostic output.

`--jsonpath=`*`FILENAME`*

The hardware blob path.

`--dtbpath=`*`FILENAME`*

The devicetree blob path.

This option is currently Arm only.

`--skhheap=[`*`ADDRESS,`***`]`***`NUMBER[b|M|G]`*

The LynxSecure Separation Kernel Hypervisor heap size and optionally, address. The heap provides for dynamic memory allocations. Dynamic memory is allocated by the hypervisor during startup only, but not after the hypervisor has begun executing subjects. The minimum required size of the heap depends on two things: the user-defined configuration and the target hardware specifics. At this time, Autoconfig Tool doesn't have the ability to estimate the required heap size; instead, it uses a default value, which should be sufficient for most configurations. Should the default value be insufficient, please increase the heap size using this option. After a successful startup, LynxSecure Separation Kernel Hypervisor prints the actual final heap utilization. That information and this option may be used to trim the heap to the minimum required size.

`--srpformat=`*`FORMAT`*

The SRP format string to pass to `mkcv -f`. See `mkcv -?` for more information.

`--srploadaddr=`*`HPA`*

The SRP load address.

`--ht`, **`--no-ht`**

Enable/disable HyperThreading (if available).

`--iommu`, **`--no-iommu`**

Enable/disable IOMMU support. Refer to Chapter 8, "*Physical Device Assignment to Subjects*" (page 57) for more details.

`--runtimediag=[none|fatalonly|all]`

Enable LSK hypervisor Run-Time diagnostics (`fatalonly` means diagnose only subject-fatal faults, such as the Triple Faults on the x86).

`--modules=MODULES`

Modules to enable in the SRP, comma-separated. Please refer to section "Enabling Additional Modules" in *LynxSecure 6.3.0 Advanced Configuration Guide* for more details.

`--smidisable`

Attempt to disable System Management Interrupts. This option only works if the target system's chipset is known to LynxSecure. Most BIOS'es lock the general SMI control in the enabled state. If that is the case, LynxSecure will disable some individual SMI sources instead.

This option is x86-only.

`--tickspersec=`*`NUMBER`*

Scheduling ticks per second.

`--rifargs=`*`STRING`*

Add extra arguments for RIF.

`-g`, **`-gg`**, **`-ggg`**

Increase the verbosity of the tool. Useful when debugging the issues with the tool, not needed most of the time. For example, `-gg` also enables debugging printouts.

Creating a Simple SRP Image

LynxSecure Development Workflow Step 6

Using the HCV as an input, the configuration compiler generates the System Runtime Package. This binary contains the hypervisor along with any configured guest resources. The System Runtime Package is generally referred to as the SRP.

At this point, it is possible to create a simple SRP. The following example creates a single subject with all SATA, graphics, keyboard, mouse, USB, audio and network devices assigned to it. Adjust the command line below to remove any devices that are not present in the target hardware:

```
$ autoconfig target-name \
    --subject-fv=SATA*,VGA,GRAPHICS*,KEYBOARD,MOUSE,USB*,AUDIO*,NET*
```

This command creates an SRP named `target-name.srp` in the `$ENV_PREFIX/build` directory. If you are using PXE to boot the target, copy this image into the `/var/lib/tftpboot` directory. The example PXELINUX configuration described in section "PXELINUX Configuration" (page 8) may need to be edited to reflect the SRP filename. For other boot methods, refer to Chapter 5, "*Booting SRPs*" (page 27).

A detailed description of the subject specification syntax is provided in later chapters.

When the SRP is booted on the target, it boots from the target's CD/DVD-ROM (if present) or hard drive. It is now possible to proceed to installing the guest operating systems onto the target.

It is also possible to boot from an ISO image or disk image accessible over network. Doing this requires creating a virtual block device as explained in Chapter 13, "*Virtual Block Device (only x86 platform)*" (page 81).

NOTE: Pre-existing installations of the operating systems on the target may or may not be able to boot depending on the configuration of the guest operating system.

CHAPTER 5

Booting SRPs

LynxSecure® Development Workflow Step 7

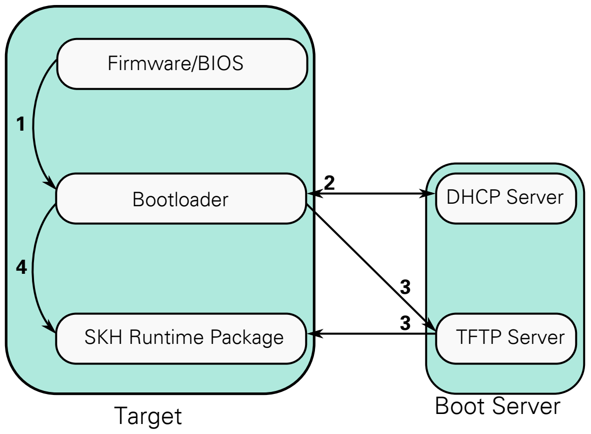
Once the SRP bootable image is generated by the configuration compiler, the bootable image must then be loaded onto the target platform. The SRP can be booted from many media formats including a Local Hard Disk, Flash Media, USB Media, or the network.

Booting LynxSecure involves the board firmware (such as the BIOS on x86-based systems), at least one boot loader and the system runtime package which finally runs on the system. The firmware loads a bootloader from either the local storage media (for example, a hard disk) or from the network, and executes it. The bootloader then boots the SRP.

Network Boot

Booting LynxSecure over network can be done using a network-capable bootloader supporting the DHCP and TFTP protocols or the PXE protocol (on x86). Note that the PXE protocol utilizes DHCP and TFTP internally.

Figure 5-1: Network Boot



Legend:

1. The firmware or BIOS loads and starts a network-capable bootloader (the exact procedure is platform- and bootloader-dependent).
2. The bootloader sends a DHCP query out and sets the IP address, gateway, and DNS.
3. The bootloader downloads the SRP from the TFTP boot server.
4. The bootloader jumps to the start of the SRP.

Please follow these guidelines to set up the corresponding service:

- section “Enabling TFTP” (page 7)
- section “Configuring and Running the DHCP Server” (page 7)

Network Boot Using PXE (only x86 platform)

PXE booting LynxSecure requires a bootloader called PXELINUX, which is part of the Syslinux family of bootloaders. Please see the section “PXELINUX Configuration” (page 8) for configuration instructions. The SRP must be in either the `raw` or the `efi` format for this boot method, which is the case for the x86 platform by default.

The example PXELINUX configuration file provided with LynxSecure (`$ENV_PREFIX/tools/buildroot/custom/board/autoconfig/pxelinux.cfg`) has an entry in the boot menu to boot the SRP; the SRP image is expected to be available at the `/var/lib/tftpboot/target.srp` path. Edit the `pxelinux.cfg` configuration file to change the location. A simple PXELINUX configuration to boot an SRP named `target.srp` without the boot menu is shown below:

```
DEFAULT mboot.c32 target.srp
SERIAL 0x3f8 115200
```

Network Boot Using U-Boot (only Arm platform)

The U-Boot bootloader can be used to boot a LynxSecure SRP over network. The SRP must be in either the `u-boot-fit` or the `u-boot-legacy` format for this boot method. The default SRP format for the Arm® platform is `u-boot-fit`.

NOTE: The U-Boot binary supplied by the board manufacturer may be configured to only support the legacy image format. If that is the case, please either rebuild U-Boot with FIT image support or use an `mkcv` option to override the default format.

Assuming that the DHCP and TFTP services have been configured (see section “Development Host Services Configuration” (page 6)), the following sequence can be used to boot a LynxSecure SRP:

1. Switch on the target.
2. Interrupt autoboot by pressing any key during the countdown.
3. Enter the following commands:

```
setenv serverip 10.20.31.1
setenv bootfile filename.srp
setenv srp_net_boot "dhcp; tftpboot; bootm;"
setenv bootcmd "run srp_net_boot"
saveenv
```

4. Power cycle the target, and it should now boot automatically from the tftp server.

Various features of the target board and the local network may require additional configuration commands, such as setting the gateway IP, the Ethernet MAC address, etc. Please refer to the U-Boot and board documentation for these advanced topics.

LynxSecure has the following requirements for the U-Boot configuration on the Arm platform:

- U-Boot must be configured to pass control to the booted software on the boot CPU at Exception Level 2 or 3 (EL2 or EL3).
- If the target uses the `spin-table` CPU enable method, U-Boot must be configured to pass control to the booted software on enabled CPUs at Exception Level 2 (EL2).

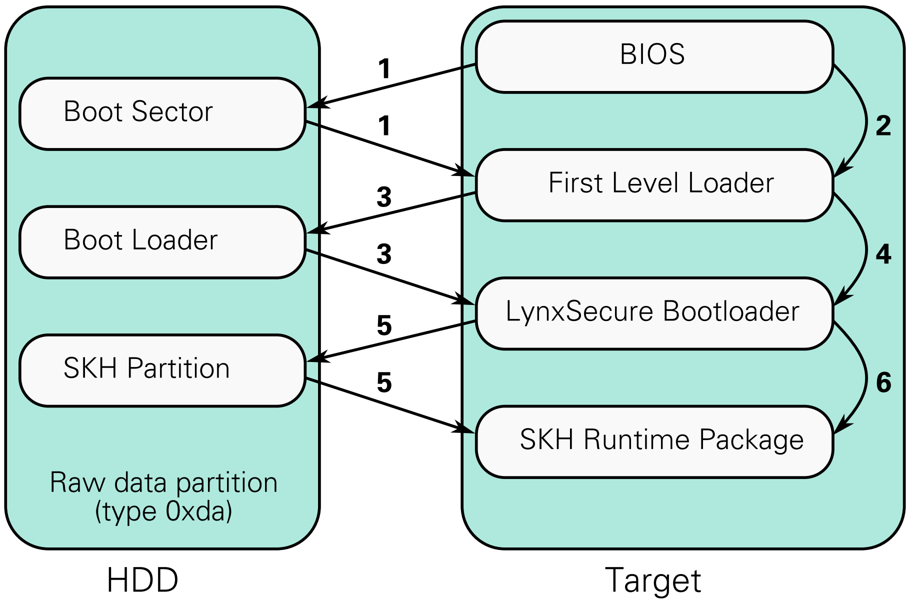
These requirements may require rebuilding of the manufacturer-supplied U-Boot binary. LynxSecure includes a properly configured U-Boot binary in the software package for the target board, if one is necessary.

NOTE: It is recommended to use the U-boot bootloader binary that comes with LynxSecure for the particular target board. This binary is properly configured to boot LynxSecure.

Boot from a Mass Storage Device Using LynxSecure Bootstrap Loader (only x86 platform)

NOTE: The information in this section applies only if the target platform is x86.

Figure 5-2: Booting SRP Using LynxSecure Bootstrap Loader



Legend:

1. BIOS loads data from the hard disk’s raw data partition into the target memory.
2. BIOS jumps to the start address of the first level bootloader.
3. First level bootloader loads data from the hard disk’s raw data partition into the target memory.
4. First level bootloader jumps to the start address of the LynxSecure bootstrap loader.
5. LynxSecure bootstrap loader loads data from the hard disk’s raw partition into the target memory (LSK runtime).
6. LynxSecure bootstrap loader jumps to the start address of SRP.

Installing to Hard Disk (only x86 platform)

Hardware Discovery Linux® comes with a tool to install an SRP to the target's storage device (HDD, USB stick, etc.).

1. Boot the target with the Hardware Discovery Linux. Refer to section “PXELINUX Configuration” (page 8), section “Booting the Hardware Discovery Linux from a CD/DVD-ROM or USB Storage (only x86 platform)” (page 9).
2. Create an SRP on the host machine (refer to Chapter 7, “*Subjects*” (page 45)), then copy the SRP to the target machine.
3. Login to the Hardware Discovery Linux.
4. Partition the storage device. Refer to section “Creating Partitions on the Storage Device” (page 11).
5. `$ install-srp -d DEVICE -f SRP-FILENAME`

Installing to USB Stick (only x86 platform)

1. Connect the USB stick to the development machine. Determine the device name for the newly connected disk from the `dmesg` output or by looking at the new symbolic links appearing in the `/dev/disk/by-id` directory.
2. Run the `install-srp` script:

   ```
   $ install-srp -d NEWLY-CONNECTED-DEVICE -f SRP-FILENAME
   ```
3. Remove the USB drive.
4. Re-install the USB drive into the target system.
5. Power-up the target system, enter the BIOS setup and choose the USB stick as the first bootable device. Save the BIOS settings.

NOTE: The above method can be used to install the SRP onto the target's hard drive. After shutting down the target, remove the hard drive and connect the disk to the development host (e.g. using a USB-to-SATA adapter).

To connect a SATA disk directly to the development machine's SATA controller, the development machine needs to be powered down. It will need to be powered down again when removing this disk from the development machine as well.

Configure the LynxSecure Bootstrap Loader (only x86 platform)

The LynxSecure Bootstrap Loader can be configured using the LynxSecure CDK's `install-srp` script. It supports the following options:

-h, --help

Display the list of the supported options.

-a *SECS*, **--autoboot** *SECS*

Set delay for automatic boot of the SRP to *SECS* seconds. By default, the bootstrap loader boots the SRP immediately. Pressing a key during the automatic boot delay will take the user to the bootloader's console shell.

-c *CMDLINE*, **--cmdline** *CMDLINE*

Set command line arguments passed to the SRP to *CMDLINE* string. See section “Passing Options to RIF from a Custom Bootstrap Loader” (page 34) for the list of options recognized by the SRP.

-d *DEVICE*, **--devname** *DEVICE*

Install the SRP to the *DEVICE* device.

The script deletes the partition table if the *entire* block device (e.g. `/dev/sda`) is used as the SRP location. During the SRP installation, the script will create a partition called `lsk_boot` which will keep the SRP image. The script will show a warning message and will ask for the user confirmation before installation:

```
$ user@hostname:~$ install-srp -d /dev/sdc -f ./target.srp
```

```
WARNING: Using the entire device /dev/sdc! All data on '/dev/sdc' will be destroyed!
Do you want to proceed? (y/n):
```

The partition table won't be wiped out if *`DEVICE`* points to a partition (e.g. `/dev/sdc3`).

NOTE: To upgrade an SRP, do not use the entire device as an argument to this option; use a partition where the SRP has been previously installed.

-f *`FILE`*, **--filename** *`FILE`*

Use *`FILE`* as the SRP image.

-P *`PROMPT`*, **--prompt** *`PROMPT`*

Set the prompt displayed in the bootloader's console to *`PROMPT`*.

-s *`PORT[,BAUD[,CLOCK]]`*, **--serial** *`PORT[,BAUD[,CLOCK]]`*

By default, the output of the bootloader is sent to the standard COM1 serial port (0x3f8, 115200 baud, 1.8432MHz clock) and the VGA terminal. A different serial port can be configured using this option; the argument specifies the new base port number (e.g. `-s 0x2f8`), optionally followed by the baudrate (e.g. `-s 0x1030,19200`), optionally followed by the clock rate (e.g. `-s 0x1030,19200,1843200`).

-t, --trusted-mode

Enable trusted boot. Refer to section “Booting SKH Runtime Package Using the Trusted Initialization” in *LynxSecure 6.3.0 Advanced Configuration Guide* for additional information on trusted initialization.

Boot LynxSecure Using Syslinux Boot Loader (only x86 platform)

NOTE: The information in this section applies only if the target platform is x86.

Syslinux boot loader can be used to boot LynxSecure, it can be installed to disk using the LynxSecure CDK or Hardware Discovery Linux. For more information regarding Syslinux, please refer to the http://www.syslinux.org/wiki/index.php/SYSLINUX.

The following example shows how to boot multiple SRPs using Syslinux. Assume the two SRPs `1st.srp` and `2nd.srp` need to be installed onto the disk `/dev/sdb`.

- Create a clean GUID Partition Table (GPT), if it is not created already.

```
$ sgdisk -Z /dev/sdb
```

```
$ sgdisk -og /dev/sdb
```

- Create a 1 GB partition.

```
$ sgdisk --new 1::+1G -c 1:lsk /dev/sdb
```

- Mark that the GPT partition `/dev/sdb1` is going to be bootable.

```
$ sgdisk -A 1:set:2 /dev/sdb
```

- Create a FAT filesystem on the bootable partition `/dev/sdb1`. On the CDK host:

```
$ mkfs.vfat /dev/sdb1
```

At the Hardware Discovery Linux shell prompt:

```
$ mkfs.fat /dev/sdb1
```

- Install syslinux to the partition `/dev/sdb1`

```
$ syslinux -i /dev/sdb1
```

- Install `gptmbr.bin` to the disk `/dev/sdb`:

```
$ cat /usr/share/syslinux/gptmbr.bin > /dev/sdb
```

- Copy the following file to the partition `/dev/sdb1`:

```
$ mount /dev/sdb1 /mnt/
$ cp /usr/share/syslinux/menu.c32 /mnt
$ cp /usr/share/syslinux/mboot.c32 /mnt
```

At the Hardware Discovery Linux shell prompt, copy the following additional files:

```
$ cp /usr/share/syslinux/libutil.c32 /mnt
$ cp /usr/share/syslinux/libcom32.c32 /mnt
```

- Copy SRPs to the partition `/dev/sdb1`:

```
$ cp 1st.srp 2nd.srp /mnt
```

- Create the file `syslinux.cfg` on `/dev/sdb1`, for example:

```
DEFAULT menu.c32
prompt 0
timeout 60
MENU TITLE LynxSecure

LABEL FirstSRP
menu label FirstSRP
kernel mboot.c32 1st.srp

LABEL SecondSRP
menu label SecondSRP
kernel mboot.c32 2nd.srp
```

- Boot the target from the disk `/dev/sdb`, a menu shall be presented with the `FirstSRP` and `SecondSRP` options. Choose one of the options to boot.
- Some BIOS may fail to boot with an error message "`Invalid partition table`". In such scenario either set the MBR boot flag on `/dev/sdb1` using `fdisk` on CDK Host or use an alternate boot method using LynxSecure bootloader.

  The following example shows setting MBR boot flag using `fdisk`. Assume the disk appears as `/dev/sdb` on the CDK host.

```
$ fdisk /dev/sdb

Command (m for help): a
Partition number (1-4): 1

Command (m for help): w
The partition table has been altered!
```

Boot LynxSecure Using UEFI (only x86 platform)

NOTE: The information in this section applies only if the target platform is x86.

LynxSecure supports UEFI non-secure boot mode. The target system BIOS "Secure Boot Control" should be disabled so that LynxSecure can boot in UEFI non-secure boot mode. The SRP generated by the `mkcv` utility can be executed directly by UEFI firmware.

Preparing UEFI SRP Boot Disk

The physical disk should use a GUID Partition Table (GPT). We suggest to use a minimum of 1GB for the LynxSecure disk partition.

1. Boot Autoconfig Tool image on the target and log in.

   It is also possible to perform this procedure on the development host by disconnecting the target's hard drive as described in section “Installing to Hard Disk (only x86 platform)” (page 29); in that case, the `mkfs.fat` should be replaced with `mkfs.vfat`.

2. Inspect the mount table using the `mount` command and ensure that the target disk is not currently in use. If the disk is currently in use, unmount the devices in question by executing `umount`.

3. Initialize the GPT on the disk:

   ```
   $ sgdisk --clear -g /dev/sdX
   ```

   Here and below, *`sdX`* is the name assigned by the OS to the disk where the SRP is being installed, such as `sda`, `sdb`, and so on. If this command reports that it was unable to update the kernel's partition table, please reboot the system before continuing.

4. Use `sgdisk` to create a new partition with type code `ef00` (EFI System) named `lsk`.

   ```
   $ sgdisk -n 1::+1G -c1:"lsk" -t 1:ef00  /dev/sdX
   $ sgdisk -n 2::+1G -c2:"vds" -t 2:8300  /dev/sdX
   ```

   Additional partitions may be created in this step. For example, create another new partition with code `8300` (Linux System) named `vds` that may be used for VDS configuration persistent storage.

   ```
   $ sgdisk -n 3::+30G -c3:"fv0" -t 3:8300 /dev/sdX
   $ mkfs.fat /dev/sdX1
   $ mount /dev/sdX1 /mnt
   $ mkdir -p /mnt/efi/boot
   $ cp target.srp /mnt/efi/boot/bootx64.efi
   $ umount /mnt
   ```

Newer BIOS'es allow selecting the boot file from any directory and add it to the boot menu. Some older BIOS'es with UEFI support need the file in a specific directory with a specific name; for example, `bootx64.efi` file in the `/efi/boot` directory. Place the SRP according to the BIOS capabilities.

Preparing UEFI SRP Boot Using PXE

Please follow the guidelines in section “PXELINUX Configuration” (page 8) to setup a configuration for booting SRP over UEFI PXE protocol. Replace `target.efi` inside `/var/lib/tftpboot/grub.cfg` with the appropriate SRP name.

Boot LynxSecure Using kexec command on Linux (only x86 platform)

LynxSecure SRP can be booted from Linux using the `kexec` command. Command line options to the SRP can be specified with the `--command-line` option of the `kexec`. See section “Passing Options to RIF from a Custom Bootstrap Loader” (page 34) for the list of options recognized by the SRP.

1. Generate the SRP in elf32 format using the `--srpformat="elf32"` option of the Autoconfig Tool.
2. Boot Linux and get a shell prompt. Below, we assume that the filesystem path to the SRP file is `/tmp/test.srp`.

3. Load and boot the SRP using following commands:

```
$ kexec -l -t multiboot-x86 --command-line="multiboot_memmap" /tmp/my.srp
$ kexec -e
```

NOTE: Since the graphics driver in linux has already initialized the graphics card, VGA output from SRP won't be displayed while booting SRP using kexec. Serial output will be displayed as expected.

Boot LynxSecure from a SD card (only Arm platform)

NOTE: The information in this section applies only if the target platform is Xilinx® Zynq® Ultrascale+™ MPSoC.

Steps for booting a SRP from a SD card using Uboot bootloader:

1. Prepare a SD card using instructions from: https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842385/How+to+format+SD+card+for+SD+boot
2. Copy BOOT.BIN to a FAT32 formatted SD card as the first file placed on it.
3. Boot the target and execute the following commands at the Uboot prompt:

```
$ setenv loadaddr "0x10000000"
$ setenv kernel_img "test.srp"
$ setenv cp_kernel2ram "mmcinfo && fatload mmc 0 ${loadaddr} ${kernel_img}"
$ setenv sdcardboot "run cp_kernel2ram && bootm ${loadaddr}"
$ setenv bootcmd "run sdcardboot"
$ saveenv
```

Passing Options to RIF from a Custom Bootstrap Loader

The Runtime Initialization Function (RIF) is the first LynxSecure component to execute when LynxSecure starts. RIF boot parameters can be set either via the Multiboot protocol [http://www.gnu.org/software/grub/manual/multiboot/multiboot.html], or as SRP built-ins. For more information on setting built-in RIF parameters, see the Autoconfig Tool documentation.

`trusted_init`

Enables RIF support for trusted initialization. See the section “Booting SKH Runtime Package Using the Trusted Initialization” in *LynxSecure 6.3.0 Advanced Configuration Guide* for details.

`quiet`

Silences non-error messages from early RIF startup code.

`debug`

Enables additional debug information for early RIF startup.

`multiboot_memmap`

Normally, RIF attempts to discover the system memory map using one of the following methods, in the decreasing priority order:

- Using INT15h (AX=E820h) service, as described in the ACPI specification.
- From the Multiboot information block, using the `mmap_*` fields.
- From the Multiboot information block, using the `mem_*` fields.

This option makes RIF forgo the INT15h method and only use the Multiboot information block to discover the system memory map. This can be useful to work around BIOSes that do not support INT15h (AX=E820h) service, or return incorrect information to that request.

This option is x86-only.

`early_serial=`*`[compatible:]port[,baud[,clockrate[,lcr]]]`*, **`early_serial=stdout,`***`clockrate`*

This option configures the serial device used for early RIF messages. RIF would only use it up to the point when it validates the configuration from the HCV; at that point, it will switch to the serial port configured in the HCV.

In case this option is not supplied, RIF will buffer the early output and then print it to the consoles configured in the HCV.

On the x86 platform, the *`compatible`* string is ignored; the x86 version of LynxSecure only supports 16550-compatible serial devices at this time. The *`port`* parameter is required and must specify the hexadecimal address of the I/O port to use for serial output. By default, RIF startup code outputs its messages to port 0x3f8 (COM1), at 115200 baud and 1.8432MHz clock.

NOTE: The 16550-compatible serial devices divide the clock by 16 internally for the baudrate generation. E.g., the default 1.8432MHz clock corresponds to a device with a maximum baudrate of 115200.

On FDT platforms such as Arm, the *`port`* parameter may contain either a hexadecimal address, or a Devicetree path or alias for the serial device. If a numeric value is used for the *`port`* parameter, the *`compat`* string shall also be provided as defined in the Devicetree specification [https://github.com/devicetree-org/devicetree-specification/releases/download/v0.2/devicetree-specification-v0.2.pdf]. If a Devicetree path or alias is used for *`port`*, the *`compat`* string is ignored. The *`baud`* and *`clockrate`* parameters are mandatory on FDT platforms.

The second form of the option is only applicable to FDT platforms; it makes RIF honor the console setting selected by the `stdout-path` property of the `/chosen` node in the Devicetree.

The line control settings are specified as the number of data bits, followed by parity, followed by the number of stop bits. The supported values for parity are `n` (no parity), `o` (odd parity), `e` (even parity), `m` (mark/one parity) and `s` (space/zero parity). On both x86 and Arm platforms, the default setting for line control is `8n1` (8 data bits, no parity, 1 stop bit).

`early_vga`

On the x86 platform, this option enables the graphical console (VGA or EFI framebuffer) for early RIF messages. If not supplied, RIF will buffer the early output and print it to VGA or framebuffer later if they are configured in the SRP.

`usb_handoff`

Forces LynxSecure to perform USB BIOS-to-OS handoff prior to starting subjects. Use this option to work around USB-related System Management Mode BIOS bugs causing, for example, non-functional USB controllers in subjects. This option is x86-only.

`no_usb_handoff`

In some situations, LynxSecure performs USB BIOS-to-OS handoff automatically prior to starting subjects. Use this option to prevent that operation in case it is causing problems. This option is x86-only.

`paniccall=`*`map_start,map_size,entry_point`*

Configures the location for transfering control during a panic in LynxSecure. The address range specified by *`map_start`* and *`map_size`* is mapped read-only and executable; the location must be reported as reserved (either by the system memory map provided by the firmware, or in the system memory map in the HCV). The address specified by *`entry_point`* gets called during a panic; it is passed a single argument, a pointer to a string describing the panic (NUL-terminated, possibly truncated to some undefined size). The argument is passed according to the SysV ABI function calling sequence requirements, as applicable to the platform on which LynxSecure executes. The entry point may get called by multiple CPUs. If the entry point returns, SKH will complete the panic sequence normally.

For this option to work, the `paniccall` module must be linked into both RIF and SKH binaries. Refer to section “Enabling Additional Modules” in *LynxSecure 6.3.0 Advanced Configuration Guide* for information on configuring the RIF/SKH modules.

`ignore_builtin_args`

Specifying this option causes any RIF parameters built into the SRP to be ignored. This option is useful if RIF contains built-in parameters that are undesirable, but building a new SRP is inconvenient. This option is needed if it is desired to override any built-in parameters as built-in parameters take priority over those passed from the bootloader.

Segmented Boot Using U-Boot (only Arm platform)

Overview

By default LynxSecure SRPs includes the necessary images to boot each subject. While this is convenient in many cases, it comes with a drawback. If the guest images need to be changed, then the SRP must be re-created. Configuring your system to perform a “segmented boot” provides a path to avoid this limitation. A segmented boot is one in which the SRP and guest images are loaded separately.

This section describes booting Linux using a segmented boot on the NXP® S32V234™ MPSoC hardware. The kernel and ramdisk referenced are the result of building the default NXP S32V234 MPSoC Buildroot configuration. A similar process is possible on all supported hardware.

Normal Boot Process

To understand all of the configurable options for segmented boot, it is best to first start with an explanation of the normal boot procedure:

1. U-Boot loads the SRP into RAM at a specific HPA and boots it. In this case, all of the boot images are included in the SRP so they are already available to the later stages of the boot process.
2. RIF initializes the system. The bootable guest images are remapped into the appropriate subject address spaces. For example, the guest image “fv0.LINUX_KERNEL” at HPA 0xC1000000 might be remapped to GPA 0x03080000. This is usually a read-only mapping.
3. Control is handed to SKH, which then launches all configured subjects.
4. Each FV subject starts by executing the emulation layer. This layer relocates the bootable images remapped in step two, and places them at a location that is calculated based on the type of the bootable image. This relocation is performed for two reasons. First, most operating systems overwrite the memory that they are booted from. By relocating the image, a pristine copy is preserved that can be used when the subject is rebooted. Second, most kernels have requirements related to their boot address. Relocating the kernel at this stage allows for those requirements to be met automatically.
5. Finally, the guest image is booted. If the guest is rebooted, the process repeats starting at step three.

Components of a Segmented Boot

1. SRP - This must be re-configured to *not* include guest images for any subject that will be segmented. Instead memory is reserved for these guest images.

2. Boot Trailers - these are small binaries that contain configuration data for each image. They are necessary to reliably boot using this method.
3. Bootloader - The bootloader must be configured to load not just the SRP, but also the needed guest images and boot trailers.

Configuring an SRP

Starting with a normal SRP:

```
autoconfig mksrp s32v \
--subject-fv1=kernel=linux-kernel-s32v234evb,\
ramdisk=linux-ramdisk-s32v234evb.img \
ram=208m,cpus=4,serial
```

This SRP will directly contain the kernel and ramdisk described. To configure subject fv1 for segmented boot the kernel and ramdisk options must be replaced with the “bootregion” option. The bootregion option describes a guest image without actually including it in the SRP.

```
kernel=linux-kernel-s32v234evb

becomes

bootregion=c1000000:fe000000:8m:LINUX_KERNEL:RO
```

The above option configures the SRP with the following:

1. A new memory region is defined for the subject of size 8 MiB
2. The HPA of that mapping is 0xC1000000. This is the address where U-Boot will load the guest image.
3. The GPA of the mapping is 0xFE000000. This is the physical address of the mapping from the guests perspective.
4. The type of the bootable image is LINUX. This tells FVS how to handle the image during boot. For LINUX type images, the Linux boot protocol is observed.
5. Finally, the image is mapped into the guest read-only. This means that the guest cannot interfere with the contents of the image and that subject level reboot will be reliable.

In a similar manner, the ramdisk is also removed from the SRP, and the new autoconfig command line becomes:

```
autoconfig mksrp s32v \
--subject-fv1=bootregion=c1000000:fe000000:8m:LINUX_KERNEL:RO, \
bootregion=c2000000:ff000000:8m:LINUX_KERNEL:RO, \
ram=208m,cpus=4,serial
```

Configuring Boot Trailers

With the guest images removed from the SRP, there must be a way to pass parameters describing the guest images to LynxSecure from the bootloader. This is accomplished using a boot trailer. These are small binaries containing information that is used to configure the boot process within FVS.

These binaries can be generated using the included `lsk_boot_trailer` tool, or they could be generated at boot time by a custom bootloader. The parameters that must be configured will vary from guest to guest. For Linux, the only required parameter is the size of the guest image:

```
$ lsk_boot_trailer linux-kernel-s32v234evb \
-o linux-kernel-s32v234evb.trailer
$ lsk_boot_trailer linux-ramdisk-s32v234evb.img \
-o linux-ramdisk-s32v234evb.img.trailer
```

The trailer format and contents are further documented in the LynxSecure API guide.

Configuring U-Boot

U-Boot must be configured to place all needed images at the correct HPA. In this case the following load addresses are used:

Image	Load Address	Notes
SRP	0x8307FFC0	
Linux Kernel	0xc1000000	As configured in the bootregion option
Linux Kernel Trailer	0xc1800000	0xc1000000 + 8MiB
Linux Ramdisk	0xc2000000	As configured in the bootregion option
Linux Ramdisk Trailer	0xc2800000	0xc2000000 + 8MiB

The boot trailer must be placed just after the configured size of the boot region. This is the location that FVS will scan for the trailer. With the above images loaded, your SRP can now be booted.

Execute In Place

In some configurations it may be desired to execute the guest image in-place. That is, instead of first relocating then booting the guest image the guest image is directly booted without relocation. This can reduce boot time and memory consumption, but will most likely preclude subject-level reboots. There are several complications related to execute-in-place:

1. The boot image's GPA must be appropriately aligned in accordance with the guest's boot protocol. For Linux, the entry point of the image must be a certain number of bytes offset from a 2 MB boundary.
2. For Linux, the ramdisk must be placed within 512 MiB of the kernel.
3. Some guests, including Linux, require that their entry point be in unreserved read-write RAM.
4. In the case of Linux, the entry point must be within a contiguous chunk of memory large enough to uncompress the kernel image.

With the above restrictions in mind, the following must be done to boot a Linux image in-place.

1. The U-Boot header should be stripped off. This is not strictly necessary, but it makes alignment easier. The U-Boot header is 64 bytes at the beginning of the binary.

```
$ tail -c +65 linux-kernel-s32v234evb > linux-kernel-s32v234evb.stripped
$ tail -c +65 linux-ramdisk-s32v234evb.img > linux-ramdisk-s32v234evb.img.stripped
```

2. The bootregions are configured as RW_RAM instead of RO.
3. The bootregions are increased in size to accommodate the decompression during early boot. In this case, the boot regions were grown to 32 MiB. Note that because these will be be R/W and not reserved, they will be used as general purpose ram after the guest boots. For that reason, the value to the "ram=" option was reduced by 64 MiB. This leaves the guest with the same total RAM allocation.
4. GPAs were adjusted to account for alignment and placement requirements.
5. HPAs were adjusted to account for the new image sizes.

The new autoconfig command line becomes:

```
autoconfig mksrp s32v \
--subject-fv1=bootregion=c1000000:80000:32m:LINUX_KERNEL:RW_RAM, \
bootregion=c4000000:03080000:32m:LINUX_KERNEL:RW_RAM, \
ram=144m,cpus=4,serial
```

The boot trailers must be configured with the XIP flag set by passing `-i` to `lsk_boot_trailer`. Setting this flag causes FVS to skip relocation:

```
$ lsk_boot_trailer linux-kernel-s32v234evb.stripped \
-o linux-kernel-s32v234evb.stripped.trailer -i
$ lsk_boot_trailer linux-ramdisk-s32v234evb.img.stripped \
-o linux-ramdisk-s32v234evb.img.stripped.trailer -i
```

Finally, U-Boot's load addresses must be adjusted to match the above autoconfig command line:

Image	**Load Address**	**Notes**
SRP	0x8307FFC0	
Linux Kernel	0xc1000000	As configured in the bootregion option
Linux Kernel Trailer	0xc3000000	0xc1000000 + 32MiB
Linux Ramdisk	0xc4000000	As configured in the bootregion option
Linux Ramdisk Trailer	0xc6000000	0xc4000000 + 32MiB

CHAPTER 6 Installing Guest Operating Systems

LynxSecure® Development Workflow Step 8

Now that the SRP is running on the target system, the guest operating systems can be configured and installed. While LynxSecure provides flexibility in this area, it is recommended that guest operating systems are installed from within a running instance of LynxSecure. This will ensure that all of the guest's resources are securely separated, and that no unauthorized resources will be configured during installation.

Installing a Fully Virtualized (FV) Guest Operating System (OS) onto the LynxSecure virtual machine is similar to installing it onto a bare-metal system. This chapter discusses common installation techniques, and highlights some of the additional capabilities that LynxSecure brings to the table.

Generic Installation

Generic Installation allows a user to install a fresh Guest OS onto a dedicated disk or a virtual disk partition. As mentioned previously, this process mirrors how a Guest OS installs natively. For instance, installation process for Window 8 on a bare-metal system would be similar to installing it on LynxSecure.

Guest OS Installation on a Dedicated Disk

A virtual Guest OS can be installed directly onto a dedicated disk. The user is not required to partition the disk prior to creating a LynxSecure SRP.

Installation procedure:

- Create a LynxSecure SRP with the desired configuration.
- Load the installation media into the target system.
- Boot up the target system and load the SRP.
- Follow the screen prompt to complete the installation.

The following example assigns graphics, audio, keyboard, mouse, and the SATA controller to the subject, which includes the hard disk and CD/DVD-ROM connected to that controller:

```
$ autoconfig mksrp target \
  -subject-fv1=SATA0,cpus=4,ram=4g,VGA0,GRAPHICS0,AUDIO0,KEYBOARD0,MOUSE0 \
  --output=/var/lib/tftpboot/target.srp
```

NOTE: If the installation does not continue after a restart of the guest, you may need to reset the whole system.

When the installation completes, ensure all device drivers are properly installed. Some Guest OS may not support all devices, and will require the user to download the drivers from the vendor and install them manually.

Guest OS Installation on a Virtualized Disk

Another option for installation is onto a Virtualized Disk. For this option, the user is required to partition the disk prior to creating a LynxSecure SRP.

- Create Virtual Disk partition using the Hardware Discovery Linux® set up GPT labels, if necessary. Refer to section “Creating Partitions on the Storage Device” (page 11).
- Update the Autoconfig Tool data base. Refer to section “Collecting Target Information on x86 Platform” (page 15).
- Create a LynxSecure SRP with the desired configuration.
- Load the installation media into the target system, if necessary. The installation media for the Guest OS installation can be a physical CD/DVD media or an ISO image that can be installed over NFS or FTP. Pls refer to section “Sample Configurations” (page 83) for additional configuration information of NFS/FTP.
- Boot the SRP on the target system.
- Follow the on screen prompt to complete the installation.

KDI Installation

LynxSecure has the capability to embed the KDI images into an SRP or to load it from the network via iPXE. The user must create a bootable image file first. For instance, for the LynxOS®-178 2.3.x, refer to *LynxOS-178 2.3.x Board Support Guide* on how to create the KDI.

Pre-requisite:

- Hardware for the target has been collected
- Bootable KDI images
- SRP has been created with the proper configuration

Installation procedure:

After the KDI image is loaded, the user needs to follow the Guest OS installation procedure based on the image they have prepared. The following example loads the LynxOS-178 bootable image from the SRP or loads it from the network onto ramdisk:

```
$ autoconfig mksrp target \
  -subject-vds0=password=root123,loginconsoles=ttyS0 \
  -subject-fv1=virthd=Los178,initparams="CD:3 BEV:1 DISK:4 KDI:2",boottimeout=30,\
ram=4g,VGA0,KEYBOARD0,MOUSE0,SERIAL0,virte1000=NET0,\
orompath=$ENV_PREFIX/build/iPXE-ROM/8086100e.rom,kdipath=los178.img \
--output=/var/lib/tftpboot
```

Restart the System

LynxSecure Development Workflow Step 9

Once the guest operating systems have been installed, LynxSecure must be fully rebooted by safely shutting down each guest and resetting the whole system.

The system can now be restarted to allow LynxSecure and all configured Guest operating systems to boot in the desired configuration. The users can operate the installed Guest operating systems similarly to their natively installed instances.

CHAPTER 7 Subjects

A subject is a collection of resources exported by the LynxSecure® Separation Kernel Hypervisor, and it is an entity with its own code executing within the protective framework of the LynxSecure Separation Kernel Hypervisor.

Please refer to section “Subjects” in *LynxSecure 6.3.0 Architecture Guide* for more details.

LynxSecure supports the following types of subjects:

- Fully Virtualized (FV) Subjects.

  Fully virtualized subjects simulate the bare hardware platform behavior.
- Para-virtualized (PV) subjects.

  Para-virtualized subjects run operating systems and other software that has been created or modified specifically to run under LynxSecure.

  For instance, LynxSecure uses a special PV Linux® subject to support device emulation for FV subjects. Please refer to Chapter 9, “*Virtual Device Server (only x86 platform)*” (page 65) for more details.
- LynxSecure Application (LSA).

  A LynxSecure Application (LSA) is a subject created entirely by the developer; it is a program that runs directly on the virtual platform provided by LynxSecure.
- Built-in Test (BIT) Subjects.

  The BIT subject runs the set of tests contained in the BIT Suite module.
  - Initiated Built-In Test (IBIT).

    A set of tests which will run at system startup. IBIT presence is mandatory.
  - Continuous Built-In Test (CBIT).

    A set of tests which will be continuously cycled through in the background on each processor after the system is booted. CBIT presence is optional.

NOTE: On Arm® targets, there is no support for:

- Virtual Device Server (VDS).
- Virtual Devices that require VDS.
- PV Linux subjects.

The Autoconfig Tool syntax is different for different subject types.

Generic Subject Autoconfig Syntax

Synopsis

```
autoconfig mksrp target-name \
  --subject-[TYPE]NAME= \
    type=TYPE,... \
    isa=[64bit|32bit],... \
    initparams=PARAMETERS-LIST,... \
    initialstate=[running|stopped],... \
    ram=SIZE[K|M|G],... \
    ropagesize=SIZE[K|M|G],... \
    cpus=NUMBER,... \
    pat[=[true|false]],... \
    sanpath=FILENAME,... \
    bootregion=HPA:GPA:SIZE:TYPE:MODE[:NAME],... \
    memregion=NAME:SIZE[:[MODE][:TYPE]],... \
    guestimage=NAME:FILENAME[:TYPE],... \
    wbinvd=[performance|native|secure_fast_dma|secure_fast_cache_flush],... \
    perm=[readsched|writesched|halt|restart|guesthypercall|maintenance|skdb|absclock
        |readaudit|writeaudit],... \
    subjperm=SUBJECT-NAME:start|stop|restart|suspend|resume|status|all-subject-states
        |flexdonate,... \
    realtime[=[true|false]],... \
    cpu_rng[=[true|false]],... \
    userhypercalls[=[true|false]],... \
    breakonstart[=[true|false]],... \
    irq=IRQ:FROM-SUBJECT-NAME,... \
    audit.irq=IRQ|auto,... \
    vcpu.cache-capacity=NUMBER%|SIZE|BITMASK,... \
    [VIRTUAL-DEVICE0,VIRTUAL-DEVICE1,...],... \
    [PHYSICAL-DEVICE0,PHYSICAL-DEVICE1,...],... \
    ...
  --sched.maintenance=SUBJECT-NAME1[,SUBJECT-NAME2,...] \
  --sched.focus=NUMBER% \
  --sched.minorframe=TICKS \
  --sched.policy=NAME:SUBJECT-NAME1[,SUBJECT-NAME2,...] \
    ...
```

Subject Options

`--subject-[`*`TYPE`*`]`*`NAME`*`=[type=`*`TYPE`*`][,isa=[64bit|32bit]]`

Adds a new LynxSecure subject *`NAME`* of the type *`TYPE`* and using the Instruction Set Architecture *`ISA`* to the configuration.

The subject's type can be specified either by using a prefix in the subject name (e.g. `fvWindowsXP`, `fv0`, etc.) or by using a `type=`*`TYPE`* option. The Instruction Set Architecture specification is optional; if not specified, the tool finds it out from the subject image, or uses the default one for the architecture.

Table 7-1: Autoconfig Tool Subject Type Syntax

Subject Type	LynxSecure Subject ISA	Supported Target Platform	Description
`fv`	`64bit` or `32bit`	x86 and Arm	FV subject. The ISA is ignored on the x86 platform, because subject code may switch between the instruction sets freely, but it is useful on the Arm platform where the subject can't.
`vds`	`64bit` or `32bit`	x86	64-bit or 32-bit PV Virtual Device Server subject; determined automatically from the PV Linux kernel image being used.
`pvlinux`	`64bit` or `32bit`	x86	64-bit or 32-bit PV Linux; determined automatically from the PV Linux kernel image being used.
`pvlsa`	`64bit` or `32bit`	x86 and Arm	64-bit or 32-bit PV LSA subject; determined automatically from the image being used.
`pvlynxos`	`32bit`	x86	PV LynxOS subject.

Special subjects (IBIT and, if requested, CBIT) are created automatically by the Autoconfig Tool.

The special Virtual Device Server subject with the reserved name `vds0` is a PV Linux subject that will be created automatically by Autoconfig Tool (refer to Chapter 9, "*Virtual Device Server (only x86 platform)*" (page 65)), if virtual device emulation is required. The name `vds0` cannot be used for regular subjects. The Virtual Device Server uses dedicated kernel and ramdisk images provided with LynxSecure.

NOTE: Lynx Software Technologies, Inc. doesn't provide a 32-bit LSA library for the Arm platform; customers should use the 64-bit LSA library in all cases.

`initparams=``*PARAMETERS-LIST*`

Override the default `initparams` for the subject. For FV subjects, the supported parameters are described in section "FV Subject Autoconfig Syntax" (page 51). For a PV Linux subject, this is the Linux kernel command line.

`initialstate=[running|stopped]`

The initial state of the subject.

`ram=`*`SIZE[K|M|G]`*

Explicitly define the total usable subject RAM size.

`ropagesize=`*`SIZE[K|M|G]`*

Explicitly specify the subject ROPAGE size.

ROPAGE region contains a list of resources exported to the subject. By default, the region size is predefined to 64Kb.

If a lot of resources are assigned to a subject, such as virtual devices, physical devices, memory regions, guest images, etc., the Configuration Tool may fail to create an SRP. In this case, manual modification of the ROPAGE size may be required.

cpus=*NUMBER*

Explicitly define the number of virtual CPUs for the subject.

pat[=[true|false]]

Enable the Page Attribute Table feature. See Intel® IA-32e Software Developer's Manual for the description of this feature.

This setting exists for security reasons. Care should be taken when allowing subjects to use the PAT feature: this feature lets subject software access device memory with the Write-Combining memory type, which freezes some CPUs. This results in all subjects running under LynxSecure being frozen until system reset. A malicious subject may use it to disrupt the operation of other subjects running on the same system. This is a hardware issue with no solution other than identifying and removing the flow to the problematic device or not allowing untrusted subjects to use the PAT feature.

Note that disallowing subjects to use the PAT feature may have a negative effect on performance of some of those subjects.

This option is x86-only.

sanpath=*FILENAME*

Specify a memory sanitization image. Please refer to section “Memory Sanitization” in *LynxSecure 6.3.0 Architecture Guide* for more details.

bootregion*=HPA:GPA:SIZE:TYPE:MODE[:NAME]*

Specify a custom boot memory region for a subject. This region should be pre-populated with a valid image before the subject is started. HPA and GPA can be either an address or `auto'. Addresses must be specified as a hexadecimal number starting with 0x. SIZE accepts decimal, hexadecimal, or decimal postfixed with K, M, or G. TYPE is one of LINUX_KERNEL, LINUX_RAMDISK, CPIO, or KDI. MODE can be one of RO, RW_RAM, or RW_RESERVED. In the case of RW_RAM, the guest image is not marked as reserved, and it is likely that the guest operating system will modify the contents of the memory region. Specify RW_RAM only if subject restart is not needed.

Bootregions are useable in conjunction with the boot trailers. See section “Configuring FVS boot parameters at run time” in *LynxSecure 6.3.0 API Guide* for more information.

Table 7-2: Autoconfig Tool Boot Region Syntax

Parameter	Description
HPA	The host physical address of the memory region. This value is usable by external tools to preload a memory region with a bootable image. This value must be aligned on a 4 KB boundary.
GPA	The guest physical address of the memory region. This value controls the mapping of the HPA to GPA. This value must be aligned on a 4 KB boundary. If an arbitrary GPA is acceptable, this can be specified as "auto".
SIZE	The size of the memory region. This should be specified to be at least large enough to hold the desired bootable image.
TYPE	This option controls how the given image is processed during initialization.
MODE	Mode can be one of RO, RW_RAM, or RW_RESERVED. The RO mode creates a boot region that is not modifiable by the guest. Boot regions described in this manner will allow the subject to be restarted reliably. The RW_RESERVED mode allows the subject to modify the boot image, but the memory range is not allocated for general purpose RAM. The RW_RAM mode does not reserve the memory region and

Parameter	Description
	the guest operating system is free to use that memory region as general purpose RAM. The RW_RAM mode will most likely preclude subject restart as the boot image is likely to be modified.
`NAME`	The name of the memory region. If this name is not provided, then the default *`SUBJECT-NAME.TYPE`* will be used. If the given name does not end in *`.TYPE`*, then *`.TYPE`* will be appended to the given region name.

`memregion=`*`NAME:SIZE[:TYPE[:MODE]]`*

Specify a custom memory region for a subject. In case of a shared memory region, the region parameters will be overridden by the last occurrence of the memory region specification. The name of the region is global and may be used to refer to the same region from different subjects. Mode can be either `RW` or `RO`. Supported types: `PROGRAM`, `SHM`, `RESERVED`, `CMA`, `DMA`. `PROGRAM` creates available RAM. `SHM` creates memory shared between subjects; this is the default type. `RESERVED` RESERVED creates a generic reserved memory region. `CMA` and `DMA` create Linux Contiguous Memory Allocator or regular DMA pools, respectively (only on Devicetree platforms).

`guestimage=`*`NAME:FILENAME[:TYPE]`*

Specify a custom guest image for a subject. Supported types: `PROGRAM`, `RESERVED`, `BOOT`.

`wbinvd=[performance|native|secure_fast_dma|secure_fast_cache_flush]`

Write Back and Invalidate Cache (WBINVD) CPU instruction handling.

This option is x86-only.

`perm=[readsched|writesched|halt|restart|guesthypercall|maintenance|skdb|absclock|readaudit|writeaudit]`

Grant subject specific permissions.

`subjperm=`*`SUBJECT-NAME`***`:start|stop|restart|suspend|resume|status|flexdonate|all-subject-states`**

Set permissions to act on another subject.

Table 7-3: Permissions

Name	Description
`start`	Permission to start the target subject.
`stop`	Permission to stop the target subject.
`restart`	Permission to restart the target subject.
`suspend`	Permission to suspend the target subject.
`resume`	Permission to resume the target subject after being suspended.
`status`	Permission to get the status of the target subject.
`flexdonate`	Permission to donate schedule time to another subject. The target subject must be scheduled to run on the same processor core as this subject.
`all-subject-states`	Grant all permissions to manage the state of the target subject. Shortcut for `start:stop:restart:suspend:resume:status`.

`realtime[=[true|false]]`

Improve the real-time response of the subject. For real-time subjects, LynxSecure programs the physical interrupt controller to honor interrupt priorities specified by the subject for its interrupts in its virtual interrupt controller. This prevents physical CPU interruptions from low priority interrupts when a higher priority interrupt is being serviced in the real-time subject. LynxSecure also reduces its schedule-driving clock interrupt priority on a physical CPU occupied by a real-time subject to the lowest priority. However, some other internal LynxSecure interrupts may still be serviced at the highest interrupt priority.

This attribute imposes some limitations on the subject. For example, a real-time subject's virtual CPUs can't share a physical CPU with virtual CPUs of other subjects. On some platforms with multiple physical interrupt controllers, real-time subjects cannot have their external device interrupts come from more than one physical interrupt controller.

`cpu_rng[=[true|false]]`

Allow/disallow access to the CPU built-in hardware Random Number Generator.

This setting exists for security reasons. It may be possible for subjects with access to this feature to use it for a covert information channel between them.

This option is x86-only.

`rdt.monitor[=[true|false]]`

Allow/disallow access to the Intel's RDT Monitoring features.

This feature is not fully virtualized. The functionality is limited to monitoring LynxSecure subject resource utilization. It is not possible to monitor any other entities. For more information about RDT refer to Intel 64 and IA-32 Architectures Software Developer’s Manual Volume 3B: System Programming Guide, Part 2, section INTEL RESOURCE DIRECTOR TECHNOLOGY (INTEL RDT) MONITORING FEATURES.

An example application is supplied with Virtual Device Server to report the resource monitoring information. For more information refer to section “Cache and Memory Bandwidth Monitoring Tool” (page 73).

This option is supported on certain x86-64 processors which support Intel's RDT Monitroing Technology.

`userhypercalls[=[true|false]]`

If set to true, hypercalls are allowed from user-level code.

Currently, this feature is only supported on the x86 platform. Guest code issuing a hypercall must have its current privilege level (CPL) less than or equal to its I/O privilege level (IOPL); otherwise, the hypercall returns a Permission Denied error code.

This option is x86-only.

`breakonstart[=[true|false]]`

Set an external debugger breakpoint at the subject start address (VCPU #0 only).

This option is Arm only.

`irq=`*`IRQ`*`:`*`FROM-SUBJECT-NAME`*

Specify a custom IRQ number to inject into the subject. x86 valid ranges: [0..223], Arm valid ranges: [32..1019].

`audit.irq=`*`IRQ`***`|auto`**

Specify a custom audit IRQ number to inject into the subject. x86 valid ranges: [0..223], Arm valid ranges: [32..1019].

Specifying audit IRQ will also grant audit buffer read permissions to that subject.

`vcpu.cache-capacity=`*`NUMBER%`***`|`***`SIZE`***`|`***`BITMASK`*

Specify the last level cache partitioning settings for all virtual CPUs in the subject. The feature is disabled by default. Specifying the cache capacity in bytes or percents will result in an exclusive cache partition. If cache sharing between subjects is desired, use raw capacity bitmasks. All subjects without an explicit cache capacity specification will share one cache partition (the entire remaining cache capacity).

`[VIRTUAL-DEVICE0,VIRTUAL-DEVICE1,...]`

Specify a virtual device. See next chapters for more details:

- Chapter 12, “*Virtual Audio (only x86 platform)*” (page 79);
- Chapter 13, “*Virtual Block Device (only x86 platform)*” (page 81);
- Chapter 14, “*Virtual Keyboard Video Mouse (only x86 platform)*” (page 85);
- Chapter 15, “*Virtual Network (only x86 platform)*” (page 89);
- Chapter 16, “*Virtual UART (only x86 platform)*” (page 95);
- Chapter 18, “*Virtual USB (only x86 platform)*” (page 99).
- Chapter 19, “*LSA.store Full Disk Encryption (only x86 platform)*” (page 113).

`[PHYSICAL-DEVICE0,PHYSICAL-DEVICE1,...]`

Specify a physical device. Refer to Chapter 8, “*Physical Device Assignment to Subjects*” (page 57) for more details.

Global Options

`--sched.maintenance=``SUBJECT-NAME1`**`[,`**`SUBJECT-NAME2`**`,...]`**

Add specified subjects to the maintenance schedule.

`--sched.focus=``NUMBER%`

Generate in-focus operational schedule for each subject sharing the CPU core with other subjects. Enabled by default for subjects having virtual VGA.

`--sched.minorframe=``TICKS`

Setup the minimum limit for the minor frames in fixed time schedules.

`--sched.policy=``NAME:SUBJECT-NAME1`**`[,`**`SUBJECT-NAME2`**`,...]`**

Create custom fixed time schedule.

FV Subject Autoconfig Syntax

```
autoconfig mksrp target-name \
--subject-NAME=type=fv,... \
  apic=[true|false],...\
  savdebug=DBGLEVEL,...\
  fwpath=FILENAME,... \
  boottimeout=SECONDS,... \
  initparams=[HD|CD|ROM|KDI]:PRIORITY,... \
  orompath=ROMFILE,... \
  kernel=bzImage[,ramdisk=initramfs.cpio.gz],... \
  kdipath=FILENAME[,kdidst=ADDR][,jump=ADDR],... \
  fvspath=FILENAME,... \
...
```

Subject Options

`apic=[true|false]`

This options defines whether local APIC and I/O APIC emulation is provided for the subject (true) or not (false). If local APIC and I/O APIC are not enabled, the subject can only access the legacy PIC (8259A). By default, local and I/O APICs are enabled.

This option is x86-only.

`savdebug=`*`DBGLEVEL`*

Allows debugging output over the diagnostic output for FVS.

Table 7-4: The debugging levels

The debugging level	Description
0	LOG_NONE
1	LOG_PANIC
2	LOG_BANNER
3	LOG_ERROR
4	LOG_WARN
5	LOG_INFO (default)
6	LOG_DEBUG
7	LOG_VERBOSE

`fwpath=`*`FILENAME`*

Specifies a custom firmware image for booting fully virtualized subjects. By default, `$ENV_PREFIX/build/bios.bin` will be used which is legacy BIOS. To boot x86-64 UEFI based guests, specify `fwpath=OVMF.fd` and make sure that either physical LPC or virtual LPC (virtlpc) is assigned to the subject.

`boottimeout=`*`SECONDS`*

This feature controls displaying of the boot menu. The Virtual BIOS waits until *`SECONDS`* expires for user to press **F12** to display list of boot devices which allow user to choose boot device to boot from. If **F12** is not pressed within time specifed by *`SECONDS`*, the Virtual BIOS will continue with the default boot device, which is the first device by order.

`initparams=[HD|CD|ROM|KDI]:`*`PRIORITY`*

Augment the arguments passed to the virtual BIOS. In particular, this allows to control the boot device priority for a FV subject. The lower the *`PRIORITY`* number the higher is the boot priority. Specifying the boot priority is ignored for UEFI guests as boot order can be changed and saved from BIOS setup menu.

This syntax is x86-only.

Table 7-5: Supported boot devices

Name	Description
`CD`	A physical or virtual CD/DVD ROM, including USB CD/DVD ROMs.
`HD`	A physical or virtual hard disk drive, including USB mass storage.
`ROM`	Bootable option ROMs like iPXE.
`KDI`	A bootable kernel image such as LynxOS®/LynxOS-178 or Linux bzImage.

The `initparams` option also allows passing command line options to the Linux kernel embedded into the SRP, as described below under the `kernel` option.

`orompath=`*`ROMFILE`*

Adds PXE ROM image *`ROMFILE`* to the FV subject. PXE firmware is required for network boot. Most physical networking devices include PXE firmware on an Option ROM. Virtual networking devices and some physical network cards do not have built-in PXE option ROM. In these cases, iPXE ROM can be used instead. The corresponding physical or virtual ethernet adapter must be assigned to the FV subject as well.

iPXE option ROM images for virtual networking interfaces are supplied with LynxSecure. Use `8086100e.rom` for the virtual E1000 interfaces and `10ec8029.rom` for the virtual NE2000 interfaces. For instructions how to build iPXE option ROMs for other network adapters, refer to section “iPXE” in *LynxSecure 6.3.0 Open Source Software Guide*.

This option is x86-only.

NOTE: Running iPXE option ROMs under LynxSecure currently requires the CPU to support the "unrestricted guest" feature of the VT-x. Older CPUs may not have this feature even though they support VT-x.

kernel=*bzImage***[,ramdisk=***initramfs.cpio.gz]*

The subject will run the Linux kernel named in the `linuxkernel` subject parameter, and, if one is provided, the ramdisk in the `linuxramdisk` subject parameter. Both the Linux kernel and the ramdisk are included to the SRP.

When booting a FV Linux subject from the SRP, the subject's `initparams` option is passed to Linux as the kernel command line. If the delimiter string `^^` is present in the `initparams`, only the portion of the `initparams` following `^^` is passed as the kernel command line. This enables SeaBIOS- or FVS-specific options to be hidden from the Linux kernel.

kdipath=*FILENAME***[,kdidst=***ADDR***][,jump=***ADDR***]**

If a fully virtualized subject is configured to run a Kernel Downloadable Image (KDI) containing the kernel and ramdisk for the LynxOS-178 or LynxOS operating system, it is possible to configure the SRP to include the image (rather than boot it from some external storage).

If needed, the address where the KDI is loaded and the address of the kernel's entry point can be specified using the `kdidst` and `jump` options, respectively.

fvspath=*FILENAME*

The path to the FVS (FV subject firmware) image.

PV Linux Subject Autoconfig Syntax (only x86 platform)

```
autoconfig mksrp target-name \
--subject-NAME=type=pvlinux,... \
  loginconsoles=[TTY1:TTY2:...],...
  kernelconsole=TTY[:BAUD],... \
  kernelearlyconsole=TTY[:BAUD],... \
  kernel=FILENAME,... \
  ramdisk=FILENAME,... \
...
```

Subject Options

loginconsoles=*[TTY1:TTY2:...]*

Colon-separated list of TTY device names to allow logins from. By default, PV Linux subjects (including VDS) do not create login processes for any TTYs.

When a login console over a virtual UART is to be configured in a PV Linux subject, the `loginconsoles=` option needs to be supplied with the virtual device name as a tty name. The user may not know the virtual device names, however, until the HCV is created. The workaround is to create an HCV without specifying the `loginconsoles=` option and run Autoconfig Tool with a `-g` option, note the names of virtual devices for each subject and then re-run Autoconfig Tool, adding the `loginconsoles=` options with the discovered device names to the command line.

`kernelconsole=`*`TTY[:BAUD]`*

Add console settings to the Linux kernel command line.

`kernelearlyconsole=`*`TTY[:BAUD]`*

Add early console settings to the Linux kernel command line.

`kernel=`*`FILENAME`*

Specify PV Linux Kernel image.

`ramdisk=`*`FILENAME`*

Specify PV Linux Ramdisk image.

PV LynxOS Subject Autoconfig Syntax (only x86 platform)

```
autoconfig mksrp target-name \
--subject-NAME=type=pvlynxos,... \
  kdipath=FILENAME[,kdidst=ADDR][,jump=ADDR],... \
  virthd=....,
  virtnet=....,
  virtuart=....,
...
```

Subject Options

`kdipath=`*`FILENAME`***`[,kdidst=`***`ADDR`***`][,jump=`***`ADDR`***`]`**

Add a Kernel Downloadable Image (KDI) to the SRP.

`virthd=....`

Add a PV block device. See Chapter 13, "*Virtual Block Device (only x86 platform)*" (page 81) for more details.

`virtnet=....`

Add a PV network device. See Chapter 15, "*Virtual Network (only x86 platform)*" (page 89) for more details.

`virtuart=....`

Add a PV UART device. See Chapter 16, "*Virtual UART (only x86 platform)*" (page 95) for more details.

LSA Autoconfig Syntax

```
autoconfig mksrp target-name \
--subject-NAME=type=[pvlsa|fv][isa=[64bit|32bit]],... \
  lsapath=FILENAME,... \
...
```

NOTE: Autoconfig Tool creates a skeleton definition for LSA subjects that may need manual tuning for a particular configuration.

Subject Options

`lsapath=`*`FILENAME`*

Specify the LSA executable image. By default, `$ENV_PREFIX/`*`arch`*`/libexec/lsa.bin` is used.

IBIT Subject Autoconfig Syntax

```
autoconfig mksrp target-name \
--explicit=ibit \
--ibitpath=FILENAME \
--ibitfailaction=ACTION \
--ibitinitparams=PARAMETERS-LIST \
--ibitwatchdog=NUMBER \
...
```

Global Options

`--ibitpath=`*`FILENAME`*

The path to the IBIT image.

`--ibitfailaction=`*`ACTION`*

Default action on IBIT subject failure.

`--ibitinitparams=`*`PARAMETERS-LIST`*

IBIT subject init parameters.

`--ibitwatchdog=`*`NUMBER`*

IBIT watchdog timeout (ticks).

CBIT Subject Autoconfig Syntax

```
autoconfig mksrp target-name \
--cbits \
--cbitpath=FILENAME \
--cbitfailaction=ACTION \
--cbitinitparams=PARAMETERS-LIST \
--cbitwatchdog=NUMBER \
...
```

Global Options

--cbits

Enables CBIT subjects.

The CBIT subjects are not automatically created by the Autoconfig Tool. Since they are special subjects, they are not displayed in the hardware information output. Their presence can be verified by looking at the XML file.

--cbitpath=*FILENAME*

CBIT image.

--cbitfailaction=*ACTION*

Default action on CBIT subject failure.

--cbitinitparams=*PARAMETERS-LIST*

CBIT subject init parameters.

--cbitwatchdog=*NUMBER*

CBIT watchdog timeout (ticks).

Sample Configuration

```
autoconfig mksrp target-name \
  --subject-vds0=password=root123,loginconsoles=ttyS0 \
  --subject-fv1=virthd=Los178,initparams="CD:3 BEV:1 DISK:4 KDI:2",\
 boottimeout=30,ram=4g,VGA0,KEYBOARD0,MOUSE0,SERIAL0,NET0, \
    orompath=$ENV_PREFIX/build/iPXE-ROM/80861502.rom
```

This configuration describes a FV subject booting GuestOS over network using the directly assigned network device `NET0`.

CHAPTER 8 *Physical Device Assignment to Subjects*

LynxSecure® supports direct assignment of physical devices to subjects. Normally, any physical device that has hardware resources such as I/O memory, ports and/or interrupts may be assigned to a subject. There are exceptions, however. For example, PCI-to-PCI bridges cannot be assigned to subjects. Devices can only be assigned to subjects with both read and write permissions. In other words, there is no read-only device assignment.

Devices that are not directly on the system bus but behind a controller device, such as i2c endpoints, AHCI mass storage devices, etc. usually cannot be assigned to subjects individually, but only along with their controller. Assigning a controller device to a subject automatically assigns all devices behind it to the same subject. Exceptions include:

- LynxSecure supports individual PCI endpoint device assignment to subjects on the x86 platform.
- LynxSecure supports individual USB device assignment to subjects when using the Virtual USB feature (page 99).

NOTE: Any physical device can only be assigned to at most one subject.

NOTE: LynxSecure doesn't guarantee that an arbitrary direct-assigned device will be fully functional in the virtual environment provided by LynxSecure. This is determined by a number of factors, some of which are discussed in the section “Physical Device Assignment to Subjects” in *LynxSecure 6.3.0 Architecture Guide*.

Pseudo-devices

On the Arm® platform, the Devicetree often contains nodes that do not describe any physical device, but rather contain configuration information that physical device nodes refer to. In order to make this information available in the subject's virtual Devicetree, LynxSecure allows the assignment of such device nodes (referred to as "pseudo-devices") to subjects as if they were physical devices. Moreover, LynxSecure allows the assignment of a single pseudo-device to multiple subjects. By definition, LynxSecure considers a device node to be a pseudo-device if the node doesn't have any physical resources such as I/O memory or interrupts.

Using SR-IOV Devices

SR-IOV is a feature of some PCIe devices allowing for multiple Virtual Machines to share one physical device. It is achieved by the primary physical SR-IOV capable device (called the *Physical Function*, or PF) presenting a certain number of additional physical devices on the PCI bus called the *Virtual Function* devices, or VFs. These VF devices can be assigned independently to different subjects in LynxSecure.

The VF devices usually differ from the PF device, but are identical to each other. This means that the VFs need a different GuestOS driver than the PF. For example, the Linux® `igb` driver only supports the Physical Functions of a 82576 NIC; one needs to enable the `igbvf` driver to use the Virtual Functions of the same NIC.

The VFs often don't exist or can't operate properly without the SR-IOV aware device driver of the PF device being active and properly configured in the PF owner subject. Should anything happen to the PF owner subject or the PF driver, the VF devices may simply disappear and stop functioning in their respective owner subjects.

At this time, LynxSecure only supports SR-IOV Virtual Functions in paravirtualized Linux subjects.

As mentioned above, the Virtual Functions do not exist until enabled by the corresponding Physical Function. In Linux subjects (both PV and FV), Virtual Functions can be enabled/disabled on a Physical Function owned by this subject using the `manage-sriov` script. To enable all the Virtual Functions of a device, invoke the `manage-sriov` command in the guest owning the Physical Function, using the HCV name (only in PV subjects) or the bus-device-function address to identify the device:

```
# manage-sriov NET0
# manage-sriov 03:00.0
```

To enable a specific number of Virtual Functions, use the `-n` option. To disable the Virtual Functions on a device, use the `-d` option.

LynxSecure does not emulate a hot-plug mechanism for SR-IOV Virtual Functions. Because the operating systems only perform the discovery of PCI devices upon startup, they will not detect the presence of Virtual Functions unless they have been enabled prior to the subject's startup. The subjects owning Virtual Functions can be configured to be initially stopped; see section “Generic Subject Autoconfig Syntax” (page 46) for details.

Device Assignment Limitations

- FV subjects with direct-assigned DMA-capable external devices require an IOMMU for proper functioning. PV subjects don't have this limitation.

  LynxSecure provides a workaround for systems with no IOMMU, which allows at most one such FV subject. See *LynxSecure Advanced Configuration Guide*.

- Normally, the subject IRQ base is selected automatically by the Autoconfig Tool or the LynxSecure itself at boot time. The following information is only important if the configuration is created by manual editing of the HCV.

  Subjects may have specific requirements for the IRQ base. All fully virtualized subjects require an IRQ base of 32. PV subjects may need a different IRQ base. The IRQ base is static; it may not be dynamically changed by the subject.

- Embedded Graphics Considerations. A direct assignment of the on-board Intel Integrated Graphics Device to a subject may require that the LPC bridge (normally named `LPC0` by the Autoconfig Tool) is also directly assigned to the same subject.

- LynxSecure doesn't support assigning physical PCI bridges to subjects.

- LynxSecure doesn't support PCI hotplug. This includes devices with the Thunderbolt interface. Hotpluggable PCI devices can be assigned to subjects, however, as long as they are permanently connected.

- IRQ Sharing and Conflicts.

  In general, devices which share the same host IRQ may not be assigned to different subjects. However, depending on the devices' capabilities and their owner subject types, LynxSecure can use a number of techniques to resolve the conflict and let the devices be assigned to different subjects.

  The Autoconfig Tool displays information that helps to resolve IRQ sharing conflicts. Otherwise, avoiding device conflicts requires knowledge of the devices within the target system.

  – Know which devices share an IRQ.

  – Know which devices must use static PCI interrupts, i.e. are not MSI-capable or are assigned to a FV subject. For most devices assigned to a FV subject, LynxSecure can transparently emulate the device interrupt by periodically polling the device, thus removing the need for a host IRQ for that device.

CAUTION! Polling results in increased CPU utilization.

- Know which devices can (optionally) use MSI.
  - For devices assigned to PV subjects, LynxSecure may use MSI interrupts to emulate legacy interrupts.
  - For devices assigned to FV subjects, LynxSecure may use MSI-X interrupts or MSI interrupts with per-vector masking to emulate legacy interrupts. Also, LynxSecure can emulate legacy interrupts using the Command/Status registers if the "interrupt disable" bit is supported in the Command register.
  - MSI allows for a fine-grained differentiation of interrupts between devices. For example: with MSI disabled, two NICs may share the same PCI IRQ. However, with MSI enabled, they are assigned two unique (non-shared) IRQs, so the devices can be assigned to different subjects.
- Know the PCI topology/hierarchy of the target system. General topology knowledge is useful to determine which devices share interrupts, which devices can use MSI to help use non-shared interrupts, and which devices are subject to IOMMU constraints.
- Know the IOMMU constraints. The IOMMU hardware partitions DMA capable devices (if enabled). PCI devices that share a parent PCI bridge that is not transparent to the IOMMU hardware must be assigned to the same subject.

• Single-Root I/O Virtualization (SR-IOV) Limitations:
  - The host BIOS is expected to initialize the SR-IOV registers on SR-IOV capable devices. Sometimes this requires enabling SR-IOV support in the BIOS Setup menu. If those registers are not initialized, the SR-IOV capability on that device will not be available in the subject where it is assigned. If this behavior is observed, the host BIOS is the most likely culprit (and a warning to that effect is printed to the diagnostic console).
  - Multi-Root I/O Virtualization and VF Migration are not supported.
  - To use SR-IOV Virtual Function devices, their Physical Function must be assigned to a subject running a dedicated Physical Function driver.
  - If an SR-IOV Virtual Function device assigned to a subject is enabled later than that subject is started, it may remain invisible to GuestOS until the subject is restarted. We recommend setting the initial state of such subjects to Stopped and only start them after the Virtual Functions they use are enabled.
  - LynxSecure doesn't support Virtual Function assignment configurations where making the assigned VFs discoverable to guest code cannot be achieved by creating at most one virtual PCI bridge. The exact conditions where this is the case are too involved to describe in brief, but direct-assigning no more than one VF of any PF to a particular subject should work in most cases.

Autoconfig Syntax

Synopsis

```
autoconfig mksrp target-name \
  --subject-fv1=[DEVNAME][NUMBER|*][#[NUMBER|*]][?][@][:KEY1=VAL1,[:KEY2=VAL2]...],... \
    identitymem,... \
    pciecfg,... \
    devres[=[auto|host]],... \
    devicereset[=[true|false]],... \
  ...
```

```
autoconfig mksrp target-name \
  --iommu[=true|false],... \
  --explicit=[rmrr|memflows|systemram|gpa|gva|devresources|irq|ibit| \
              vdevbdf|misc|absclock],[...]],... \
  --forcepolling=DEVICE[,DEVICE...],... \
  --nopolling=DEVICE[,DEVICE...],... \
  --device-deps[=[all|ignore|refs|no-refs][,...] \
  ...
```

Subject Options

[DEVNAME][NUMBER|*][#[NUMBER|*]][?][@][:KEY1=VAL1,[:KEY2=VAL2]...]

The Autoconfig Tool assigns a name to each physical device. For PCI devices, the name is based on the device's PCI class ID. For example:

- `NET0` (Ethernet adapter #0)
- `USB4` (USB controller #4)
- `AHCI0` (AHCI controller #0)
- `IDE0` (IDE controller #0)
- `GRAPHICS0` (graphics controller #0)
- `NET2#0` (SR-IOV Virtual Function #0 of Ethernet adapter #2)
- `NET0:ipaddr=192.168.0.105/24` (Network card #0 with static IPv4 address in CIDR notation)

For non-PCI devices, the name is selected in an architecture-specific manner. For example, on the x86 platform, the names of well-known platform devices are hardcoded, while other devices are named based on their name in the ACPI tables.

It is possible to specify configuration options for some of the physical devices assigned to a PV subject. For example, the user can configure the static IP address of a given network adapter assigned to a PV Linux subject.

The user can display the list of devices that can be assigned to a subject as described in section “Displaying the Target Hardware Information” (page 18).

PCI `UARTx` devices are treated in a special way. They are represented in two ways:

- As a regular fully featured PCI device `UARTx`;
- As a set of pseudo-legacy devices `SERIALx`, one for each port on the UART.

An example output from the `autoconfig show` command shows the device as follows:

```
...
UART0          1415:9501  03:00.0   16     FV  Serial controller: Oxford Semiconductor Ltd OX16PCI954
...
SERIAL1        0000:0501                       Serial Port SERIAL1, at 0xe800 (aliasing UART0)
SERIAL2        0000:0501                       Serial Port SERIAL2, at 0xe808 (aliasing UART0)
SERIAL3        0000:0501                       Serial Port SERIAL3, at 0xe810 (aliasing UART0)
SERIAL4        0000:0501                       Serial Port SERIAL4, at 0xe818 (aliasing UART0)
...
```

This lets the user assign individual ports on the PCI UART to different subjects using the `SERIALx` aliases. Assigning both the `UARTx` device and its `SERIALx` aliases to subjects is disallowed. Also, the `SERIALx` aliases of a PCI UART device can't use interrupts; the subject must access them in polling mode.

The Autoconfig Tool tracks dependencies between devices. If a subject is assigned a device that depends on another device (including a pseudo-device), the Autoconfig Tool automatically assigns that other device and all of its descendants to the same subject.

NOTE: For the Arm platform, in order to detect device dependencies, the Autoconfig Tool requires that the Devicetree supplied with LynxSecure is used with the Autoconfig Tool `collect` command. A Devicetree blob supplied by the board manufacturer usually doesn't contain the device dependency information. A Devicetree blob containing dependency information can be built from Devicetree source using the `dtbtool` tool supplied with LynxSecure SDK.

Table 8-1: Device Assignment Syntax

Syntax	Description
`DEVICE`*`x`*	Assign `DEVICE` number *`x`*. If the number is omitted, then the first available device of that type is assigned.
`PCIDEV`*`x`*	A name used to identify PCI devices whose class ID was not recognized by the Autoconfig Tool
`DEVICE*`	All available devices of the type `DEVICE` will be assigned to the subject.
`DEVICE?`	The `DEVICE` assignment is optional for a subject; LynxSecure won't panic if the device is absent.
`DEVICE@`	Skip resetting the `DEVICE` device when its owner subject is restarted. When a subject is restarted, LynxSecure normally resets all the devices directly assigned to that subject in order to present the devices in a consistent state upon start up of the subject. Some devices, however, may not be usable after a reset if they require additional setup by the host firmware.
`DEVICE`*`x`*`#`*`y`*	Assign the SR-IOV Virtual Function number *`y`* of device `DEVICE`*`x`*. If the number *`y`* is omitted, then the first available Virtual Function is assigned. The instance of the device (the *`x`* number) cannot be omitted or set to `*` in this case. For example: `NET1#6`, `NET4#`.
`DEVICE`*`x`*`#*`	Assign the SR-IOV Physical Function (i.e. the `DEVICE`*`x`* device itself) and all its Virtual Functions. The instance of the device (the *`x`* number) cannot be omitted. For example: `NET0#*`
`GRAPHICS0`	The primary graphics card used for the VGA emulation is always the instance 0 of the `GRAPHICS` device type.
`VGA0`	The name for the legacy video device VGA.
`MOUSE0`	The name for the legacy PS/2 mouse.
`KEYBOARD0`	The name for the legacy PS/2 keyboard.
`NET0:ipaddr=192.168.0.105/24`	Assign the network adapter NET0 to a subject and assign a static address `192.168.0.105/24` to the network interface associated with the adapter. Only CIDR notation is supported.

`identitymem`

Enables the "Identity Memory Map" feature, which allows physical DMA-capable device assignment to subjects on targets with no IOMMU support. When this feature is used, RIF and SKH binaries will be configured with different load addresses, as described in section “The Host Physical Memory Layout and Customizing the Load Addresses” in *LynxSecure 6.3.0 Advanced Configuration Guide*.

`pciecfg`

Enables PCIe extended configuration space in a FV subject.

This option is only relevant to FV subjects and is ignored for other subject types.

This option is x86-only.

`devres[=[auto|host]]`

If set to `host`, allocate device resources in the subject at their host locations, if possible. This feature helps running unmodified software in a LynxSecure subject that has interrupt and other resource information hardcoded for bare hardware.

`devicereset[=[true|false]]`

Allow the subject to reset physical devices assigned to it using a generic mechanism, such as the FLR for PCI devices. If not specified, defaults to yes (true).

Global Options

`--iommu`, `--no-iommu`

Enable/disable switch for IOMMU. By default, target's support for IOMMU is determined automatically by autoconfig.

`--explicit=[rmrr|memflows|systemram|gpa|gva|devresources|irq|ibit|vdevbdf|misc],[...]]`

Enable/disable explicit specification of RMRRs, memory flows, system RAM, guest physical addresses, guest virtual addresses, physical device resources, IRQs, IBIT subjects, virtual device PCI bus/device/function address, miscellaneous items respectively. If this option is specified without any value, everything is explicit.

`--forcepolling=DEVICE[,DEVICE...]`

The list of devices which will use polling for interrupt emulation. Using this option will unnecessarily degrade devices' performance; it is not recommended except for testing purposes.

`--nopolling=DEVICE[,DEVICE...]`

The list of devices for which interrupt emulation by the means of polling is disallowed.

`--device-deps[=[all|ignore|refs|no-refs][,...]`

Specify the treatment of device dependencies when assigning devices to subjects.

Sample Configurations

No Virtual Devices

```
autoconfig mksrp target-name \
  --subject-fv1=cpus=4,ram=4g,AHCI0,AUDIO1,KVM,NET0
```

Description

This configuration describes one Guest OS with direct assign graphic, VGA, Ethernet, keyboard, mouse and disk controller.

User needs to install the device drivers manually if the Guest Operating system does not support the devices by default. For example, directly assigned NVIDIA® graphics device requires vendor-provided driver to be installed.

Several Graphics Cards

```
autoconfig mksrp target-name \
  --subject-fv1=cpus=4,ram=8g,IDE1,VGA,GRAPHICS1,AUDIO2,USB*,NET0 \
  --subject-fv2=cpus=4,ram=8g,AHCI0,GRAPHICS0,AUDIO1,MOUSE,KEYBOARD,NET1
```

Description

This configuration describes two FV subjects with directly assigned devices.

The two FV subjects will come up on two separate monitors, as well as getting assigned their own mouse, keyboard, audio, and Ethernet.

Identity Memory Map

```
autoconfig mksrp target-name \
  --subject-vds0=password=root123,loginconsoles=ttyS0,NET0,SERIAL0 \
  --subject-fv1=virthd=Win7,virtcd=CD-0000001F2-2.0.0.0-0,\
    VGA0,GRAPHICS*,MOUSE0,KEYBOARD0,LPC0,identitymem,ram=1.8g,virte1000 \
  --subject-fv2=virthd=CentOS,virtcd=CD-0000001F2-2.0.0.0-0,ram=1.8g,virte1000 \
  --no-iommu
```

Description

The example above is for a target system that does not support Intel® Virtualization Technology for Directed I/O or which has such support disabled in the BIOS.

The first Guest OS will display on the monitor and second Guest OS will be headless.

CHAPTER 9 *Virtual Device Server (only x86 platform)*

The VDS (Virtual Device Server) is a special subject running PV Linux® to assist the device emulation. The Virtual Device Server may be easily customized by the System Integrator to add or remove functionality.

This chapter provides the Autoconfig Tool syntax and describes VDS tools.

NOTE: The VDS is only supported on the x86 platform in this LynxSecure® release.

Autoconfig Syntax

```
autoconfig mksrp target-name \
  --subject-vds0= \
    persistentstorage=GPT-LABEL,... \
    squashrootfs=GPT-LABEL[:FILE-PATH],... \
    squashoverlayfs=GPT-LABEL[:FILE-PATH],... \
    password=PASSWORD,... \
    kernel=FILENAME,... \
    ramdisk=FILENAME,... \
  ...
```

The user can also assign physical devices to the Virtual Device Server subject. Refer to Chapter 8, "*Physical Device Assignment to Subjects*" (page 57) for more details.

Please refer to Chapter 11, "*Virtual ACPI (only x86 platform)*" (page 77) for more details on configuring Virtual ACPI.

Please refer to Chapter 15, "*Virtual Network (only x86 platform)*" (page 89) for more details on configuring the virtual network.

Subject Options

`persistentstorage=``GPT-LABEL`

GPT label for enabling persistent storage inside the VDS subject.

`squashrootfs=``GPT-LABEL[:FILE-PATH]`

Optional parameter that specifies the storage location of the VDS rootfs image.

`squashoverlayfs=``GPT-LABEL[:FILE-PATH]`

Optional parameter that specifies the storage location of the VDS overlay image.

`password=``PASSWORD`

Set the super user (root) password.

`kernel=``*FILENAME*`

VDS kernel image.

`ramdisk=``*FILENAME*`

VDS ramdisk image.

`keeplogs``*[=SIZE]*`

Specifies that logs stored in VDS should not be rotated, and should be allowed to grow to up to the size specified in the parameter.

VDS Console

VDS is a subject running PV Linux and it provides several ways to access the shell. The access needs to be configured at the SRP creation time.

Serial Console

The VDS subject should have access to the physical UART (refer to section “Autoconfig Syntax” (page 59)). Additionally, the VDS subject should have `loginconsoles=`*`[TTY1:TTY2:...]`* configuration option (refer to section “PV Linux Subject Autoconfig Syntax (only x86 platform)” (page 53)). For example, to enable the login prompt on the standard COM1 port (I/O port 0x3F8), add `loginconsoles=ttyS0` to the VDS subject specification.

The target running an SRP needs to be connected to the host machine by a serial cable.

Then, then user needs to start the serial communication program on the host machine (such as `minicom` or `PuTTY`) once the SRP is booted on the target system, e.g.:

```
[user@hostname ~]$ minicom -D /dev/ttyUSB0
```

VGA Console

The user may want to have the VDS shell prompt on the monitor connected to the target system. In this case, the graphics adapter should be assigned to the VDS subject. To enable login prompt on the VGA console, add `loginconsoles=tty0` to the VDS subject specification.

If the target system has only one graphics adapter and SRP should contain several FV subjects having virtual VGAs, it's still possible to access the VDS console by using a virtual KVM keystroke. Please refer to Chapter 14, “*Virtual Keyboard Video Mouse (only x86 platform)*” (page 85) for more details.

SSH

The remote access is useful when the user cannot physically access the target.

The VDS subject in the SRP should have access to the physical NIC (refer to section “Autoconfig Syntax” (page 59)).

The user needs to configure the network properly and know the *`IP-ADDRESS`* of the NIC assigned to the VDS subject.

Once the SRP is booted, the user can log in to VDS remotely using SSH client. For instance, if the host machine is running Linux, then it is simple as:

```
[user@hostname ~]$ ssh root@IP-ADDRESS
```

Virtual Device Server Tools

Virtual Device Server image includes several Lynx Software Technologies, Inc. tools for managing system settings and tracking system status.

VDSCLI

The VDSCLI (the VDS Command Line Interface) is a command line tool for configuring the Virtual Device Server system functions, such as: Persistent Storage, Global System Management, Guest Specific Management, Wired Ethernet Management, etc.

Table 9-1: VDSCLI Commands

Command	Description
`deconfig`	Remove persistent configuration.
`exit`	Leave the interactive VDSCLI shell.
`fastrestart`	Quickly and unsafely restart the system, that is without telling the subjects to shut down and waiting for them. All subjects are stopped without notification, which may result in data loss. This command is equivalent to pressing the reset button.
`fastshutdown`	Quickly and unsafely shut down the system, that is without telling the subjects to shut down and waiting for them. All subjects are stopped without notification, which may result in data loss. This command emulates pressing and holding the power button.
`help`	The `help` command can be invoked with or without an argument. If invoked without an argument, it displays a list of all VDS CLI commands with a very brief description. This helps the user to find all available commands and their purpose. If the user needs more details on how a particular command could be used, he/she needs to specify the command as an argument, for example: `help ifconfig`.
`ifconfig`	Displays the current state of wired network interfaces when there are no arguments. If there are arguments, the specified network interface is configured using the `ip addr` system command and this configuration is saved to the persistent storage.
`lidclose`	Close virtual lid by sending a "lid has been closed" message to the subject. This command emulates closing a notebook lid.
`lidopen`	Open virtual lid by sending a "lid has been opened" message to the subject. This command emulates opening a notebook lid.
`powerbutton`	Send a "power button has been pressed" message to a subject. This command emulates pressing the power button. Operating systems are normally configured to shut down once they receive this message.
`sleepbutton`	Send a "sleep button has been pushed" message to the specified subject. This command emulates pressing the sleep button. The subject's reaction to this event is normally configurable in its guest operating system.
`reset`	Factory reset, restores the default settings from the SRP. Reformats the partition specified by `config=argument` for `vds0` subject in the Autoconfig command line. This partition is used as a storage for all configuration data needed for `vds0` subject. It includes the settings (managed by the `set` command), network interface configuration. Once the partition is reformatted, the

Command	Description
	current settings are saved. They can be updated by the `set` command.
`restart`	This command sends a "power button pressed" event to the other subjects, waits until they report that they have shut down, and then restarts the system. The timeout length given to the subjects to shut down is configured by the `vacpitimeout` setting. If the timeout expires and any subject is still running, the operation fails. Use the `fastrestart` command instead to restart without waiting for the subjects.
`savelogs`	Save all system logs to the specified destination. The CLI command `show` displays various logs, all of them could be saved and sent over ssh using this command.
`schedule`	Display or switch to a new scheduling policy. In case when KVM-in-focus feature is enabled or custom XML containing several scheduling policies is used, this command allows the user to switch between schedules in runtime. When the command is used without arguments, a list of all defined schedules will be displayed; passing an argument (schedule's name or schedule ID) will lead to switching policy.
`set`	Displays all current variable values when no argument is given. With one argument, the variable value is cleared. With two arguments, the variable value is set to the specified value. If persistent storage is enabled, the persistent storage for the variable is updated.
`show`	Allows the user to inspect system information and logs. If the command is invoked without an argument, it will display a list of subcommands with their description. For example, `show route` shows the system routing tables.
`shutdown`	This command sends a "power button pressed" event to the other subjects, waits until they report that they have shut down, and then shuts down the system. The timeout length given to the subjects to shut down is configured by the `vacpitimeout` setting. If that time expires and any subject is still running, the operation fails. Use the `fastshutdown` command instead to shut down without waiting for the subjects.
`subjectrestart`	Restart the subject. The subject will behave as if a reset button has been pressed. The subject is not stopped gracefully, which may result in data loss.
`subjectresume`	Resume a suspended subject. The subject continues execution from the point where it was suspended. This command can be used only for subjects that are currently suspended.
`subjects`	Displays the list of configured subjects and their execution states. Each subject may be running, stopped or suspended.
`subjectstart`	Start a stopped subject. The subject starts as if after a power cycle.
`subjectstop`	Stop a running subject. This command is equivalent to turning off the power to that subject. The subject is not allowed to shut down gracefully, which may result in data loss. It can be used only for subjects that are currently running.
`subjectsuspend`	Suspend a running subject. The subject will stop executing until it is resumed. This command can be used only for subjects that are currently running.

Subject Monitor Administration Tool

The Virtual Device Server has a special tool to watch the FVS state and status information and manage some FVS specific settings at run-time.

Display the List of Supported Commands

Synopsis

```
monitoradm SUBJECT-NAME [help|?]
```

Description

Display the complete list of all currently supported commands in the FVS instance.

Display FVS Version Numbers

Synopsis

```
monitoradm SUBJECT-NAME version
```

Description

Display version numbers for the FVS instance.

Example

```
root@vds0:~# monitoradm fv1 version
Build Id: ENGINEERING
Build Date: 08/31/2015
Build Time: 10:32:20
URL:         svn://txx.lynx.com/lynxsecure/trunk/lynxsecure/src
Revision(s): 10971M
Built by:    username@buildhost
HCV Schema Version 0022
BCV Struct Version 0026
OK
```

Test SAV-PEF Response from the FVS

Synopsis

```
monitoradm SUBJECT-NAME ping
```

Description

Test SAV-PEF response from the FVS instance (even if the GuestOS has crashed). If the command execution hangs for a while (more than 10 seconds), that means that the FVS has crashed.

Example

```
root@vds0:~# monitoradm fv1 ping
```

```
OK
```

Display the ICV Routing Table

Synopsis

```
monitoradm SUBJECT-NAME icv
```

Description

Display the ICV routing table to see how different interrupts are handled in the FVS instance.

Example

```
root@vds0:~# monitoradm fv1 icv
#IN      type  trig IRQ GSI  intr_cnt   eoi_cnt source
  0      ACPI level  13  13         0         0 ACPI
  1      ACPI level  13  13         0         0 ACPI
  2      ACPI level  13  13         0         0 ACPI
  3      ACPI level  13  13         0         0 ACPI
  4      ACPI level  13  13         0         0 ACPI
  5      ACPI level  13  13         0         0 ACPI
  6      ACPI level  13  13         0         0 ACPI
  7      ACPI level  13  13         0         0 ACPI
  8      ACPI level  13  13         0         0 ACPI
  9      ACPI level  13  13         0         0 ACPI
 10      ACPI level  13  13         0         0 ACPI
 11      ACPI level  13  13         0         0 ACPI
 12      ACPI level  13  13         0         0 ACPI
 13      ACPI level  13  13         0         0 ACPI
 14      ACPI level  13  13         0         0 ACPI
 15      ACPI level  13  13         0         0 ACPI
 16      ACPI level  13  13         0         0 ACPI
 17      ACPI level  13  13         0         0 ACPI
 18      ACPI level  13  13         0         0 ACPI
 19      ACPI level  13  13         0         0 ACPI
 20      ACPI level  13  13         0         0 ACPI
 21      ACPI level  13  13         0         0 ACPI
 22      ACPI level  13  13         0         0 ACPI
 23      ACPI level  13  13         0         0 ACPI
 24      ACPI level  13  13         0         0 ACPI
 25      ACPI level  13  13         0         0 ACPI
 26      ACPI level  13  13         0         0 ACPI
 27      ACPI level  13  13         0         0 ACPI
 28      ACPI level  13  13         0         0 ACPI
 29      ACPI level  13  13         0         0 ACPI
 30      ACPI level  13  13         0         0 ACPI
 31      ACPI level  13  13         0         0 ACPI
 32    legacy  edge   0   2   2766006       768 PIT
 33    legacy  edge   1   1        17         6 KEYBOARD
 35    legacy  edge   3   3        10         2 COM2
 36    legacy  edge   4   4        10         4 COM1
 40    legacy  edge   8   8         0         0 RTC
 44    legacy  edge  12  12       105         0 MOUSE
 46    legacy  edge  14  14     67001         0 IDE0
 47    legacy  edge  15  15         0         0 IDE1
 48 virt INTx level  10  16        52        52 VIRTHDA_1
 49  takeover                      51         0
 50  takeover                    7228         0 VIRTLEGACYIDE0
 51  takeover                       5         0 VIRTKBDMOUSE0
 52  takeover                       0         0 VIRTUART_0
 53  takeover                      70         0 VIRTUART_1
 54  takeover                       4         0 VIRTNET_1
 55 virt INTx level  11  17      1347      1347 VIRTNET_1
 56  takeover                     117         0 VIRTNET_2
 57 virt INTx level   5  18       566       133 VIRTNET_2
 58  takeover                     452         0 VIRTNET_5
 59 virt INTx level   6  19      1558      1558 VIRTNET_5
 60 virt INTx level   7  20        23        23 VIRTUHCI0
 61  takeover                      44         0 VIRTUHCI0
 62 virt INTx level   9  21         0         0 VIRTDUMMY0
 63  takeover                       0         0 VIRTDUMMY0
 64  takeover                       1         0 MONITOR_FV1

#GSI  intr_cnt   eoi_cnt inputs
   0         0         0
   1        17         1 KEYBOARD
   2   2766006         1 PIT
   3        10         2 COM2
   4        10         4 COM1
   5       133         0 VIRTNET_2
   6      1558         0 VIRTNET_5
   7        23         0 VIRTUHCI0
   8         0         0 RTC
   9         0         0 VIRTDUMMY0
  10        52         0 VIRTHDA_1
  11      1347         0 VIRTNET_1
  12       105         0 MOUSE
  13         0         0 ACPI ACPI ACPI ACPI ACPI ACPI ACPI ACPI ACPI ACPI
ACPI ACPI ACPI ACPI ACPI ACPI ACPI ACPI ACPI ACPI ACPI ACPI ACPI ACPI ACPI
ACPI ACPI ACPI ACPI ACPI ACPI ACPI
  14     67001         0 IDE0
  15         0         0 IDE1
  16        52        52 VIRTHDA_1
  17      1347      1347 VIRTNET_1
  18       133       133 VIRTNET_2
  19      1558      1558 VIRTNET_5
  20        23        23 VIRTUHCI0
  21         0         0 VIRTDUMMY0
  22         0         0
  23         0         0
```

Display the FVS Run-Time Setting

Synopsis

```
monitoradm SUBJECT-NAME get OPTION
```

Description

Get the *`OPTION`* value from the FVS instance. See Table 9-2 (page 72) for the list of settings.

Modify the FVS Run-Time Setting

Synopsis

```
monitoradm SUBJECT-NAME set OPTION=VALUE
```

Description

Set the *`OPTION`* in the FVS instance. This command allows to set the SAV logging level of the FVS instance.

Table 9-2: The SAV Logging Level Options

Logging level option	Description
debug_level	The global SAV logging level.
ahci_log_level	The virtual AHCI device logging level.
ehci_log_level	The virtual EHCI device logging level.
ohci_log_level	The virtual OHCI device logging level.
uart_log_level	The virtual UART device logging level.
uhci_log_level	The virtual UHCI device logging level.

Example

```
root@vds0:~# monitoradm fv1 get debug_level
5
root@vds0:~# monitoradm fv1 set debug_level=6
OK
root@vds0:~# monitoradm fv1 get debug_level
6
```

Manage subject state

Synopsis

```
monitoradm SUBJECT-NAME stop|start|suspend|resume|restart
```

Description

Explicitly set the subject state; subject current state can be obtained from `/proc/lsk/subjects`.

Virtual power button

Synopsis

```
monitoradm SUBJECT-NAME button/power
```

Description

Send a virtual power button press event to the FV subject.

Virtual sleep button

Synopsis

```
monitoradm SUBJECT-NAME button/sleep
```

Description

Send a virtual sleep button press event to the FV subject.

Virtual LID event

Synopsis

```
monitoradm SUBJECT-NAME button/lid open|close
```

Description

Send a virtual LID open/close event to the FV subject.

Cache and Memory Bandwidth Monitoring Tool

The Virtual Device Server comes with a command called `monitor_app` which displays information about cache and memory bandwidth used by different subjects. Example output:

```
[root@vds0 ~]# monitor_app
SUBJECT         RMID            LLC[KB]         MBL[MB]         MBR[MB]
vds0            0               10624.0         2.4             0.0
fv0             1               1024.0          0.0             0.0
pv0             2               640.0           0.0             0.0

Press Enter to continue or Ctrl+c to exit^C
Exiting...
[root@vds0 ~]#
```

Table 9-3: CMT Fields Information

CMT Field	Description
RMID	Resource Monitoring ID is used for monitoring per subject resource utilization. Each subject gets a unique RMID.
LLC[KB]	L3 Cache Occupancy
MBL[MB]	L3 Local Memory Bandwidth
MBR[MB]	L3 Remote (ex: NUMA) Memory Bandwidth

VDS Root File System Customization and Booting

By default LynxSecure includes the VDS root filesystem in the SRP. This mode is generally the easiest to work with, but has the drawback of requiring that the system have enough ram to host the rootfs. Further, the default root filesystem included is fully customizable. LynxSecure supports several mechanisms for customizing the contents and storage location of the VDS root filesystem.

Virtual Device Server Overlays

To make simple modifications to the VDS ramdisk, the easiest supported mechanism is to use a filesystem overlay. An overlay combines the specified file trees together with the VDS root filesystem.

Once the desired file trees have been created, they can be combined together with the VDS rootfs with the following command:

```
build_ramdisk_overlay.py --tree OVERLAY-DIR --output OUTPUT
```

For example:

```
build_ramdisk_overlay.py --tree my_overlay_dir --output vds_overlay.bin
```

Once the overlay is created, it can be incorporated into the SRP by specifying the generated file as the ramdisk for the VDS subject.

Fully customized Virtual Device Server

If adding an overlay to the rootfs does not meet the applications requirements, then VDS itself can be customized. The VDS is a Buildroot based subject, and can be worked with in the same way.

- To customize the Buildroot configuration of VDS, run:

```
make -C subjects/vds/x86_64/generic/ menuconfig
```

- To build your custom vds:

```
make -C subjects/vds
```

Booting VDS from Disk

In some circumstances, it may be desired to boot VDS from disk. To achieve this, the squashrootfs option for the VDS subject is used. This option includes a minimal VDS initramfs in the SRP, which then locates the specified full rootfs and continues the boot process.

The squashrootfs option supports two modes for storing the rootfs. In the first mode, the rootfs is directly written to a named GPT partition of a disk that is available to the VDS subject. For example:

```
cat $ENV_PREFIX/86_64/libexec/vds-rootfs-x86_64.squashfs | \
        ssh root@target-ip 'cat - > /dev/disk/by-partlabel/vds_root_partition'
```

In the second mode, the squashfs is simply copied as a file within another filesystem. For example, the vds squashfs could be placed on a USB thumb drive.

CHAPTER 10 Virtual PCI-to-LPC bridge (only x86 platform)

The Virtual PCI-to-LPC Bridge feature emulates a PCI-to-LPC bridge device in FV subjects.

This virtual device is only supported on the x86 platform.

Features

- ICH9 PCI-to-LPC bridge for FV subjects.
- Provides the interrupt routing information for PCI devices at addresses 00:16.0 and [00:18.0 .. 00:1f.0]. Some GuestOS'es don't rely on ACPI tables when discovering the PCI device information, such as the interrupt routing. Instead, they try to obtain that information from the LPC bridge registers. The virtual LPC bridge is intended to resolve this issue.

Limitations

- Depends on the virtual I/O APIC presence.
- Not a full-fledged LPC bridge emulation. Only very limited functionality is supported, related to interrupt routing.
- This current status of this feature is *experimental*. Verified with the QNX GuestOS only; is known to cause issues with some other GuestOS'es.

Autoconfig Syntax

Synopsis

```
autoconfig mksrp target-name \
  --subject-fv1= \
    virtlpc,...
```

Subject Options

`virtlpc`

Add a virtual PCI-to-LPC bridge to a FV subject.

Sample Configurations

```
autoconfig mksrp target-name \
  --subject-fv0=USB*,GRAPHICS0,VGA0,SERIAL0,ram=2g,virtlpc,NET0
```

CHAPTER 11

Virtual ACPI (only x86 platform)

The Virtual ACPI feature allows the user to emulate ACPI and trigger virtual ACPI events in response to host ACPI events.

This virtual device is only supported on the x86 platform.

Features

- Power button emulation.
- Sleep button emulation.
- Battery level emulation.
- Lid open/close emulation.
- Allows intercepting a host ACPI event and performing an associated action.
- Allows sending virtual ACPI events using VDS tools.

Limitations

- The Autoconfig Tool uses heuristics to detect the host ACPI resources (combined under the artificial `ACPI0` device), because there is no standard way to enumerate these resources. These heuristics may not work on all targets.
- Hibernation is not supported.

Autoconfig Syntax

Synopsis

```
autoconfig mksrp target-name \
  --subject-vds0= \
    ACPI0,... \
  ...
```

Subject Options

`ACPI0`

Assign the artificial physical ACPI device to this subject, which allows the subject to control the host ACPI resources. Typically, this device is assigned to the Virtual Device Server subject.

`batterylevel1=`*`NUMBER`*

Defines the level 1 of battery change, measured in percentage.

`batterylevel1action=`*`ACTION`*

The action to execute when the battery discharges down to level 1.

`batterylevel2=`*`NUMBER`*

Defines the level 2 of battery change, measured in percentage.

`batterylevel2action=`*`ACTION`*

The action to execute when the battery discharges down to level 2.

`batterylevel3=`*`NUMBER`*

Defines the level 3 of battery change, measured in percentage.

`batterylevel3action=`*`ACTION`*

The action to execute when the battery discharges down to level 3.

`openlid_action=`*`ACTION`*

When the lid is opened, the specified action will be executed.

`closelid_action=`*`ACTION`*

When the lid is closed, the specified action will be executed.

`powerbutton_action=`*`ACTION`*

When the power button is pressed, the specified action will be executed.

`sleepbutton_action=`*`ACTION`*

When the sleep button is pressed, the specified action will be executed.

`acoff_action=`*`ACTION`*

When the power cable is unplugged, the specified action will be executed.

`acon_action=`*`ACTION`*

When the power cable is plugged in, the specified action will be executed.

`vacpitimeout=`*`NUMBER`*

The maximum timeout of the virtual ACPI command execution using vdscli (in seconds).

CHAPTER 12
Virtual Audio (only x86 platform)

The Virtual Audio feature allows sharing the physical audio card between several FV subjects. It is expected to be used together with the Virtual Keyboard/Video/Mouse feature.

LynxSecure® emulates one the following devices in a subject in order to support the Virtual Audio feature:

- Intel® 82801JD/DO HDA controller;
- Intel 82801 AC'97 controller.

This virtual device is only supported on the x86 platform.

Features

- A fully functional audio service - both recording and playback.
- The user is allowed to bind/unbind the audio input and output to the in-focus KVM subject at run-time using the shortcut keys, as long as the Virtual Keyboard/Video/Mouse feature is present in the configuration. Please refer to Chapter 14, "*Virtual Keyboard Video Mouse (only x86 platform)*" (page 85) for more details.

Limitations

- Only one virtual audio controller is allowed per FV subject.
- The user cannot specify the particular physical audio card to be used for audio virtualization in the Autoconfig Tool command line. The first physical audio card recognized by the Virtual Device Server will be used for recording and playback.
- Currently, only a master volume control is available for setting the recording/playback volume.
- The virtual audio quality highly depends on the CPU resources the FV subject has. Audio streaming performance may decrease depending on the number of the subjects in the overall system configuration.
- Windows® XP does not provide a driver for Intel 82801JD/DO HDA out of the box. Please use the virtual Intel 82801 AC'97 or install a driver for the Intel 82801JD/DO HDA.

Autoconfig Syntax

Synopsis

```
autoconfig mksrp target-name \
  --subject-fv1= \
    virthda,... \
    virtac97,... \
  ...
```

Subject Options

`virthda`

Assigns a virtual Intel 82801JD/DO HDA controller to a FV subject.

`virtac97`

Assigns a virtual Intel 82801 AC'97 controller to a FV subject.

CHAPTER 13 *Virtual Block Device (only x86 platform)*

The Block Device Emulation feature allows the user to have a virtual block device service for a subject. Supported in FV, PV Linux® and PV LynxOS® subjects.

LynxSecure® emulates one the following devices in a subject in order to support the Block Device Emulation feature:

- A legacy or a PCI IDE controller
- An Intel® 82801IR ICH9 Digital Office (ICH9R) Controller in the AHCI mode

This virtual device is only supported on the x86 platform.

Features

- The user may select the boot device for a subject at SRP creation time.
- The following backend media are supported:
  - A physical block device
  - A physical block device partition
  - A file on a remote FTP server (only as a CD or DVD image)
  - A file on a remote NFS server
  - A regular file located in the Virtual Device Server filesystem (modifications affect the file, if the file is located on a persistent medium)
  - A ramdisk using a regular file as the initial disk image (modifications do not affect the file)
- The HDD partition to be used as the backend medium can be identified using the following methods:
  - A GPT label (refer to section "Creating Partitions on the Storage Device" (page 11))
  - A persistent device ID created by the Autoconfig Tool (e.g. `DISK-0000001F2-0.0.0.0-5`)
  - A VDS Linux device ID (e.g. `wwn-0x50014ee65a140c43-part1`)
  - A VDS device node URI (e.g. `file:///dev/mmcblk0p3`)
  - A VDS device node name (e.g. `sr0`, `sda3`, etc.)

NOTE: Using a device node name or a device node URI is not recommended. Those names and URIs may change with the Virtual Device Server configuration changes, mostly because of the changes in the connected devices (e.g. plugging USB sticks).

- An optional LynxSecure component LSA.store can be used to provide full-disk encryption for Block Device Emulation enabled subjects that use virtual AHCI controllers. For information regarding the use and configuration of LSA.store, see Chapter 19, "*LSA.store Full Disk Encryption (only x86 platform)*" (page 113).

Limitations

- Only one virtual HBA is supported per FV subject. The user cannot use both virtual IDE and virtual AHCI controllers in the same subject.
- The Virtual HBA cannot be switched between IDE and AHCI mode at run-time.
- Virtual disk hot-plug is not supported.
- CD and DVD write operations are not supported.
- Only data CD and DVD are supported. Audio CD, Video DVD, etc are not supported.

IDE Controller Emulation

- Up to two virtual disks (CDROM or HDD).
- Only PIO transfer mode is supported.
- FTP, NFS and regular file backend devices are not supported.

AHCI Controller Emulation

- The FV subject can have up to 8 virtual disks.
- Port multiplier for the virtual HBA port is not supported.
- IDE mode is not supported.
- NCQ is not supported.
- PIO mode commands are not supported.

Autoconfig Syntax

Synopsis

```
autoconfig mksrp target-name \
  --subject-[fv|pvlynxos]= \
    virtcd=[DEVICE-NODE|FTP-URL|NFS-URL|FILE],... \
    virthd=[DEVICE-NODE|GPT-LABEL|NFS-URL|FILE],... \
    bdemode=[ahci|ide|legacy],... \
    bdetimeout=SECONDS,... \
    bdetype=[ICH9|LSK|ISA],...
```

Subject Options

virtcd=[*DEVICE-NODE***|***FTP-URL***|***NFS-URL***|***FILE***]**

Specify the virtual CD-ROM.

virthd=*DEVICE-NODE***|***GPT-LABEL***|***NFS-URL***|***FILE***]**

Specify the virtual disk.

If the *FILE* points to the file on the host machine, ramdisk will be used as a backend device.

`bdemode=[ahci|ide|legacy]`

Specify the virtual block device operation mode. Default value: 'ahci'.

`bdetimeout=``SECONDS`

Set up BDE timeout for the subject. Default value: '60'.

`bdetype=[ICH9|LSK|ISA]`

Specify the virtual block device controller. Default value: 'ICH9'.

Sample Configurations

Physical Devices

```
autoconfig mksrp target \
  --subject-fv1=virthd=wwn-0x50014ee2acc90657-part4,virtcd=sr0
```

Description

The partition `wwn-0x50014ee2acc90657-part4` will be used by LynxSecure to emulate the HDD connected over a virtual AHCI controller for FV subject `fv1`.

The physical CD/DVD drive available as `/dev/sr0` will be used by LynxSecure to emulate the CD-ROM connected over a virtual AHCI controller for FV subject `fv1`.

Regular File

```
autoconfig mksrp target \
  --subject-fv1=virthd=file:///mnt/hdd-fv1.img
```

Description

The file `/mnt/hdd-fv1.img` will be used by LynxSecure to emulate the HDD connected over a virtual AHCI controller for FV subject `fv1`.

NFS Backend

```
autoconfig mksrp target \
  --subject-vds0=eth0, \
  --subject-fv1=virthd=nfs://10.20.31.1/home/nfs/fv1.img, \
                virtcd=nfs://10.20.31.1/home/nfs/win7.iso
```

Description

The NFS file `/home/nfs/fv1.img` exported by the server `10.20.31.1` will be used by LynxSecure to emulate the HDD connected over a virtual AHCI controller for FV subject `fv1`.

The NFS file `/home/nfs/win7.iso` exported by the server `10.20.31.1` will be used by LynxSecure to emulate the CD-ROM connected over a virtual AHCI controller for FV subject `fv1`.

NOTE: The Virtual Device Server subject must have network connectivity for the NFS backend to work. A NIC must be assigned to the Virtual Device Server.

RAM Disk

```
autoconfig mksrp target \
  --subject-fv1=virthd=/home/lynxuser/ramdisks/ubuntu1010.img
```

Description

The file `/home/lynxuser/ramdisks/ubuntu1010.img` from the host machine will be used by LynxSecure to emulate the HDD connected over a virtual AHCI controller for FV subject `fv1`.

Please note that the image file will be included in the SRP.

CHAPTER 14 Virtual Keyboard Video Mouse (only x86 platform)

The Virtual Keyboard/Video/Mouse (virtual KVM) feature allows the user to share the physical keyboard, mouse and the graphics card between several FV subjects.

This virtual device is only supported on the x86 platform.

The LynxSecure® FV subject which currently holds the control over the physical graphics card, keyboard and mouse devices is called the *in-focus* subject, or the subject which has the *KVM focus*. The feature allows users to select the KVM focus subject at run-time.

LynxSecure emulates the following hardware to support the Virtual Keyboard/Video/Mouse feature:

- A PS/2 controller for keyboard and mouse virtualization;
- A VGA video card (including VESA BIOS Extensions 2.0) for video virtualization.

Features

VGA Emulation

- LynxSecure automatically selects the video mode with the highest resolution for the current video output device (monitor, projector, TV, etc.).
- LynxSecure supports a secondary video output device connected to the target. The video mode for the secondary display is also selected automatically.
- LynxSecure supports a Virtual Device Server console. The feature allows the user to log in to the Virtual Device Server at run-time and manage the target system.
- In case of Intel® graphics cards, when video RAM configured by BIOS is not sufficient to allocate the framebuffers for all configured FV subjects with a maximum resolution, the video RAM will be automatically allocated from the general purpose RAM by the Autoconfig Tool.

PS/2 Keyboard/Mouse Emulation

- Both PS/2 and USB keyboard and mouse are supported as the physical input devices. Touchscreen devices are emulated as PS/2 mouse devices for the FV subject.
- The user can plug in new USB keyboard and mouse devices at run-time. These devices will be immediately used for PS/2 keyboard and mouse emulation for the KVM in-focus subject.
- The presence of a virtual PS/2 keyboard and mouse in the configuration affects the Virtual USB. Please refer to Chapter 18, "*Virtual USB (only x86 platform)*" (page 99) for more details.
- If Virtual Audio is enabled, LynxSecure will switch the audio output to that of the KVM in-focus subject.
- LynxSecure recognizes the following keyboard shortcuts:

Table 14-1: The Virtual Keyboard/Video/Mouse System Shortcut Keys

Shortcut Keys	Description
Ctrl + Shift + F1	Switch the KVM focus to KVM subject #1.
Ctrl + Shift + F2	Switch the KVM focus to KVM subject #2.
...	
Ctrl + Shift + F10	Switch the KVM focus to KVM subject #10.
Ctrl + Shift + Up	Unpin the audio input/output; if the audio was pinned to a subject that is not the current KVM in-focus subject, the audio input/output is immediately switched to the current in-focus subject.
Ctrl + Shift + Down	Pin the audio input/output to the KVM in-focus subject.
Ctrl + Shift + Left	Switch the KVM focus to the previous KVM subject.
Ctrl + Shift + Right	Switch the KVM focus to the next KVM subject.
Ctrl + Shift + Escape	Switch the KVM focus to the Virtual Device Server console. Use the `exit` command or press Ctrl + D to exit the Virtual Device Server console and return to the last KVM subject.

Performance

The user can increase the performance of the in-focus KVM subject in case the physical CPU is shared between the in-focus KVM subject and several other subjects. In the current release, the in-focus KVM subject gets 90% of the CPU time.

The in-focus VM schedule generation is enabled by default in the Autoconfig Tool. The user has to explicitly disable it during the SRP creation if this feature is not desired.

Limitations

- LynxSecure supports up to 10 subjects with the virtual KVM.
- A subject can be configured to use either both the virtual video and the virtual PS/2 keyboard and mouse devices or the virtual PS/2 keyboard and mouse only. Virtual video without the virtual PS/2 keyboard and mouse is not supported.
- Text video modes are not fully supported. A framebuffer based console can be used by text mode subjects. To enable the framebuffer based console in Linux®, pass `vga=0x318` option on the kernel command line.
- No hardware video acceleration is provided.
- Some targets don't report the video information when using the Autoconfig Tool hardware discovery agent. The Autoconfig Tool will complain with a warning message in this case.

  Some targets may support less than 10 virtual video devices. The number of supported virtual video devices depends on the graphics framebuffer size of the physical video card.
- The Autoconfig Tool selects the physical graphics card used for the VGA emulation automatically. There's no way to select a particular graphics card for the VGA emulation from the Autoconfig Tool command line. The user has to manually modify the HCV.
- Graphics cards by VIA Technologies® and SIS® are not supported by the virtual KVM.
- Audio playback may stutter if the physical CPU core is shared and the in-focus scheduling scheduling feature is not enabled.

List of Verified Graphics Cards

The following section provides the list of graphics cards that have been successfully tested with LynxSecure in the current release.

Table 14-2: The List of Verified Graphics Cards

Device ID	Device Description
`8086:0162`	Intel Corporation Xeon E3-1200 v2/3rd Gen Core Processor Graphics Controller
`8086:0046`	Intel Corporation Core Processor Integrated Graphics Controller
`10de:0fc0`	NVIDIA® GeForce® GTX 640 OEM
`10de:0fc6`	NVIDIA GeForce GTX 650
`10de:1187`	NVIDIA GeForce GTX 760
`10de:128b`	NVIDIA GeForce GT 710
`10de:0ffe`	NVIDIA Quadro® K2000
`10de:11fa`	NVIDIA Quadro K4000
`1002:68c8`	AMD FirePro™ V4800
`1002:68f2`	AMD/ATI Cedar GL [FirePro 2270]™ 2270
`1002:6821`	AMD Radeon™ HD 8870M / R9 M270X
`102b:0534`	Matrox® Electronics Systems Ltd. G200-ER
`102b:0538`	Matrox Electronics Systems Ltd. G200-EH3
`1a03:2000`	ASPEED Technology Inc. AST2400

Autoconfig Syntax

Synopsis

```
autoconfig mksrp target-name \
  --subject-fv1= \
    virtkbdmouse,... \
    virtvga,... \
  ...
```

```
autoconfig mksrp target-name \
  --sched.focus[=FOCUS-SUBJECT-VCPU-SHARE%] \
  ...
```

Subject Options

`virtkbdmouse`

Emulates legacy mouse and keyboard in a FV subject.

`virtvga`

Assigns a virtual video device to a FV subject.

Global Options

`--sched.focus[=FOCUS-SUBJECT-VCPU-SHARE%]`

Adds special KVM-in-focus schedules for all FV subjects having virtual VGA. In this case, FV subject will have *`FOCUS-SUBJECT-VCPU-SHARE`*`%` of the CPU time.

`--sched.focus=0%` will disable KVM-in-focus schedules generation.

The Virtual Keyboard/Video/Mouse Administration Tool Syntax

The user can manage the Virtual Keyboard/Video/Mouse at run-time using the `kvmadm` tool in the Virtual Device Server.

Switch the KVM focus Subject

Synopsis

```
kvmadm setfocus SUBJECT-NAME
```

Description

Switch the KVM focus to *`SUBJECT-NAME`* subject.

Get the in-focus KVM Subject

Synopsis

```
kvmadm getfocus
```

Description

Get the information about the in-focus KVM subject, such as subject name and subject ID.

CHAPTER 15 *Virtual Network (only x86 platform)*

The Virtual Network feature allows the user to set up a virtual network connection between subjects. Both PV and FV subjects are supported. It also allows the user to share a physical NIC between subjects.

LynxSecure® supports emulation of the following NICs for FV subjects:

- NE2000 Ethernet adapter (Realtek™ 8029) for compatibility with old OSes like Windows® XP;
- E1000 Ethernet adapter (Intel® 82540EM).

These virtual devices are only supported on the x86 platform.

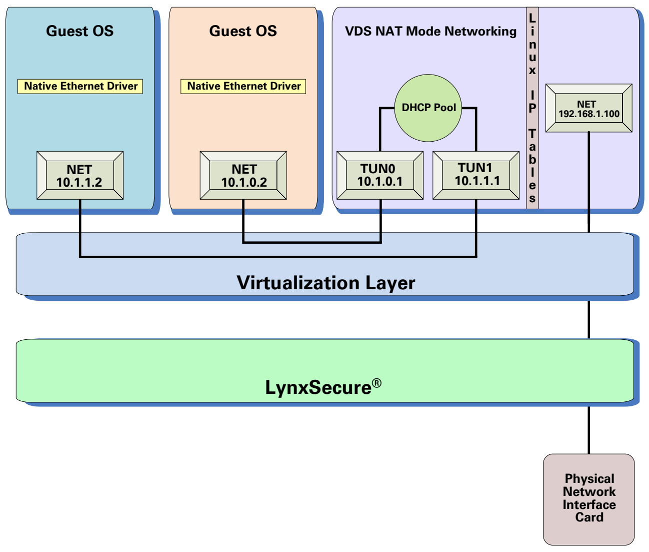
Features

- All card modes are supported including promiscuous and multicast.
- There are three type of virtual network links supported by LynxSecure:

  NAT (Network Address Translation)

  This type of a network link has the VDS operating as a server (virtual network adapters send all IP packets to the VDS). The VDS assigns IP addresses to each network link and communicates these IP addresses to client subjects using DHCP. The VDS will perform outbound NAT address translation, all packets sent outside of the system will have the VDS source IP addresses. The VDS forwards the connections through its physical network interfaces using NAT. The VDS implements the so called "masquerading" (1-to-many) NAT: it uses a single external IP address to provide network connectivity for multiple guests.

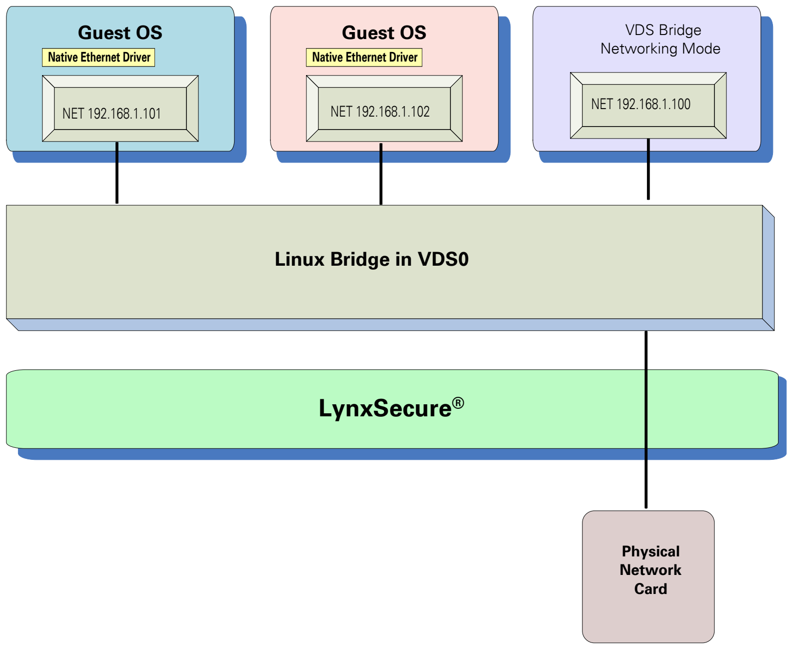
Figure 15-1: NAT



Bridging mode

With this type of a networking link, the VDS transparently forwards the level 2 (802.3) networking traffic from a subject to a physical interface. More than one subject may share the same physical interface. Each virtual Ethernet adapter has a unique MAC address. In this configuration, the VDS is not directly involved in the assignment of IP addresses, every virtual Ethernet adapter can get an IP address from the external DHCP server. The subjects' virtual network interfaces appear as distinct clients on the local network. Different subjects may be bound to different physical Ethernet adapters.

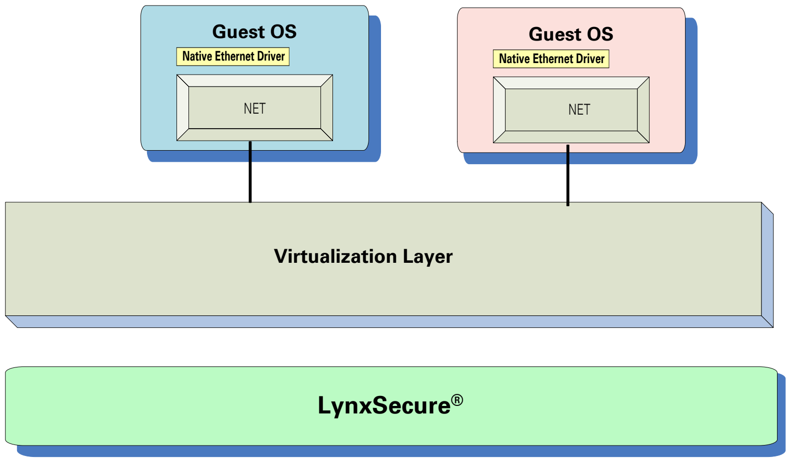
Figure 15-2: Bridging mode



Direct subject-to-subject connection

This type of networking link bypasses the VDS; two subjects have a peer-to-peer connection. This mode allows the virtual network adapters to provide a direct level 2 (802.3) link between FV subjects and/or PV Linux® subjects. It is up to these subjects to determine how the IP addresses are assigned: they may be assigned statically, or one subject may provide DHCP service for the other.

Figure 15-3: Direct subject-to-subject connection



The Autoconfig Tool can configure all these modes of networking.

- The Autoconfig Tool assigns the first supported adapter to the Virtual Device Server automatically at the SRP creation time for virtual networking in NAT mode.

Limitations

- LynxSecure supports up to 8 virtual E1000 NICs per FV subject.
- LynxSecure supports up to 4 virtual NE2000 NICs per FV subject.
- Since Windows 7 does not provide a driver for Realtek 8029 NE2000 out of the box. Please use Intel 82540EM E1000. The Realtek 8029 NE2000 driver for Windows 2000/ME/XP/2003 which also works for Windows 7 can be downloaded from the Realtek site: http://www.realtek.com/downloads.

  Since Windows XP does not provide a driver for Intel 82540EM E1000 out of the box. Please use Realtek 8029 NE2000.

Autoconfig Syntax

Synopsis

```
autoconfig mksrp target-name \
  --subject-fvX= \
    virte1000[=LEGACY-DEVICE-OPTIONS|@DEVICE-OPTIONS],... \
    virtne2000[=LEGACY-DEVICE-OPTIONS|@DEVICE-OPTIONS],... \
    virtnet[=LEGACY-DEVICE-OPTIONS|@DEVICE-OPTIONS],... \
    ... \
  --subject-pvlynxosX= \
    virtnet[=LEGACY-DEVICE-OPTIONS],... \
    ... \
  --subject-pvlinuxX= \
    virtnet[=LEGACY-DEVICE-OPTIONS|@DEVICE-OPTIONS],... \
    ... \
  --subject-vds0= \
    virtnet[=LEGACY-DEVICE-OPTIONS|@DEVICE-OPTIONS],... \
    dnsnameserver=LIST,... \
    dnsdomain=DOMAIN-NAME,... \
    dnssearch=LIST,... \
    fvnetwork=SUBNET-PREFIX,... \
    gateway=IP-ADDRESS,... \
  ... \
```

Subject Options

virte1000[=*LEGACY-DEVICE-OPTIONS***|@***DEVICE-OPTIONS***],virtnet[***LEGACY-DEVICE-OPTIONS***|@type=virte1000[@***DEVICE-OPTIONS***]]**

Add a virtual Intel Corporation 82540EM Gigabit Ethernet Controller (E1000).

virtne2000[=*LEGACY-DEVICE-OPTIONS***|@***DEVICE-OPTIONS***],virtnet[***LEGACY-DEVICE-OPTIONS***|@type=virtne2000[@***DEVICE-OPTIONS***]]**

Add a virtual Realtek Semiconductor Co., Ltd. RTL-8029(AS) Ethernet Controller (NE2000).

virtnet[=*LEGACY-DEVICE-OPTIONS***|@***DEVICE-OPTIONS***]**

Add a virtual network interface (this option is valid for PV Linux subjects only).

dnsnameserver=*LIST*

The list of DNS nameservers to use.

dnsdomain=*DOMAIN-NAME*

The DNS domain name.

dnssearch=*LIST*

The DNS domain search list.

gateway=*IP-ADDRESS*

The default gateway.

Table 15-1: Virtual Network Device Configuration Options (legacy)

Option	Description
[NAT[:A.B]|NET*x*|*SUBJECT-NAME*]	This optional argument selects one of the connection modes described above. By default, a bridged connection is created, utilizing the first NET interface assigned to the VDS. **NAT[:***A.B***]** Selects NAT connection mode. An optional subnet prefix in format *A.B* may be specified. **NET***x* Selects bridging through the specified interface. *SUBJECT-NAME* Selects the remote side of the virtual network link using a subject name.

Table 15-2: Virtual Network Device Configuration Options

Option	Description
peer=*SUBJECT-NAME*	Selects the remote side of the virtual network link using a subject name.
type=*TYPE*	Selects the virtual NIC type visible to the current subject. The parameter is applicable to FV subjects only. **`virte1000`** Intel Corporation 82540EM Gigabit Ethernet Controller (E1000). This is the default choice if `type=` is not specified in the virtual network device specification. **`virtne2000`** Realtek Semiconductor Co., Ltd. RTL-8029(AS) Ethernet Controller (NE2000).
mode=`NAT`[:*A.B*]	Selects NAT connection mode. An optional subnet prefix in format *A.B* may be specified.
physdev=*DEVNAME1*[:*DEVNAME2*:...]	Selects bridge connection mode and specifies the network devices used for bridging.
irq=*NUMBER*	Selects a custom IRQ used for this virtual network device.
bdf=`00`:*DEVNUM.FUNCNUM*	Specifies a custom PCI address used for this virtual network device. Virtual devices may reside on bus 0 only.
ipaddr=*A.B.C.D/SUBNET*	Specifies a custom IPv4 address for the given virtual network interface in CIDR notation. Applicable to the PV subjects only.
macaddr=*XX:XX:XX:XX:XX:XX*|*XX-XX-XX-XX-XX-XX*	Specifies the initial MAC address of the virtual network device. Guest OS running in a FV subject may overwite the MAC address of emulated network card (by writing data to the virtual EPPROM of the emulated card).

Virtual Network Device Configuration String Syntax (HCV)

Each virtual network device may have a null-terminated configuration string associated with this device. The string is described by the `config=` attribute in the `<virtualdeviceflow/>` tag in the HCV.

Along with the virtual device information, such as VDEVIO region and the notification interrupt (used for establishing a communication channel with the remote peer), the configuration string is exported to the subject via RO page.

The virtnet configuration string must be a space-separated list options, each option must use a format of *`key=value`*.

Table 15-3: Standard Virtual Network Device Configuration Options

Option	Description
mode=`NAT`[:*A.B*]	Selects NAT connection mode. An optional subnet prefix in format *A.B* may be specified. Applicable to the server side of the virtual network channel only.
physdev=*DEVNAME1*[:*DEVNAME2*:...]	Selects bridge connection mode and specifies the network devices used for bridging. Applicable to the server side of the virtual network channel only.
ipaddr=*A.B.C.D/SUBNET*	Specifies a custom IPv4 address for the given virtual network interface in CIDR notation. Applicable to the PV subjects only.
macaddr=*XX:XX:XX:XX:XX:XX*|*XX-XX-XX-XX-XX-XX*|auto|local|remote	Specifies the initial MAC address of the virtual network device.

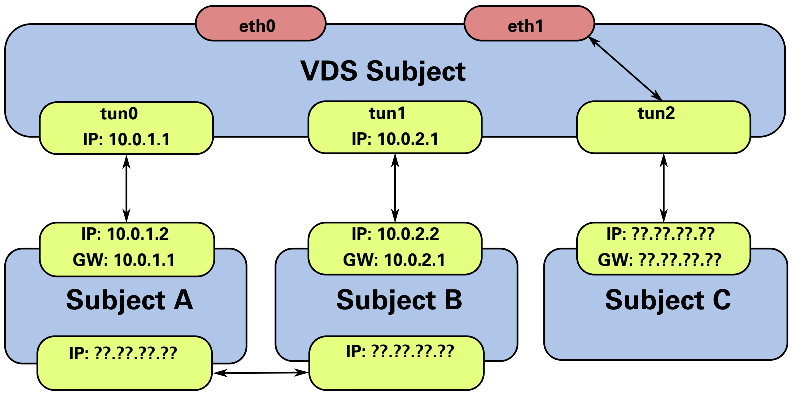
Option	Description
	Guest OS running in a FV subject may overwite the MAC address of emulated network card (by writing data to the virtual EPPROM of the emulated card). Supported values: *XX:XX:XX:XX:XX:XX*|*XX-XX-XX-XX-XX-XX* Specifies the exact MAC address. **`auto|local`** Instructs the peer owning the virtual device to generate the MAC address by itself. **`remote`** Instructs the peer owning the virtual device to request the MAC address from the remote peer. Setting this option will enforce communication with the remote peer using the shared memory communication protocol.

Sample Configuration

Here, subjects A and B have two network links using NAT mode where the VDS has assigned an IP addresses of 10.0.1.2 and 10.0.2.2, respectively. Both these links report the VDS interface as the default gateway to the subject. Subject C has a bridged networking interface; the VDS did not assign any address to that interface. Subjects A and B also have a direct connection between them.

```
autoconfig mksrp target-name \
  --subject-fv1=virthd=wwn-0x50014ee25ad749fc-part1,virte1000=NAT,virtne2000=fv2 \
  --subject-fv2=virthd=wwn-0x50014ee25ad749fc-part2,virte1000=NAT,virte1000=fv1 \
  --subject-fv3=virthd=wwn-0x50014ee25ad749fc-part3,virte1000=NET0
```

Figure 15-4: Example Virtual Network Topology



CHAPTER 16 Virtual UART (only x86 platform)

The Virtual UART feature allows the user to set up virtual serial connections between subjects by using virtual 16550a/16750 UART devices.

This virtual device is only supported on the x86 platform.

Features

- LynxSecure® supports up to 8 virtual UARTs per FV subject, 4 of them at the legacy I/O addresses.
- Creates an inter-subject communication channel. A virtual serial connection can be established between:
  - The Virtual Device Server and a FV subject
  - The Virtual Device Server and a PV subject
  - A PV and a FV subject
  - Two FV subjects

Limitations

- The Virtual UART feature does not allow sharing a physical UART between several subjects.

Autoconfig Syntax

Synopsis

```
autoconfig mksrp target-name \
  --subject-[fv|pvlinux|pvlynxos]= \
    virtuart=SUBJECT-NAME,... \
    virtuart:peer=SUBJECT-NAME \
            :type=[16550A|16750] \
            :ioport=HEX \
            :irq=PIC-IRQ \
            :config={com1|com2|com3|com4},... \
    ...
```

Subject Options

`virtuart=...`

Add a virtual UART 16550a controller to a FV subject. If the *`SUBJECT-NAME`* name is not specified, a communication channel between the Virtual Device Server and a FV subject will be created.

Sample Configurations

Let's assume the user wants to create several FV-FV serial connections.

```
autoconfig mksrp target-name \
  --subject-fv1=virthd=wwn-0x50014ee25ad749fc-part1,virtuart=fv2 \
  --subject-fv2=IDE0,virtuart=fv4,virtuart=fv1 \
  --subject-fv3=IDE1,virtuart,virtuart \
  --subject-fv4=virthd=wwn-0x50014ee25ad749fc-part4,virtuart=fv2
```

The above command line will generate the following configuration:

- A direct communication channel between a serial controller in `fv1` and a serial controller in `fv2`;
- Two serial controllers will be emulated for `fv2`. The first virtual controller will be a direct communication channel between the serial controller in `fv2` and the serial controller in `fv4`. The second controller will be a direct communication channel between the serial controller in `fv2` and the serial adapter in `fv1`.
- Two serial controllers will be emulated for `fv3`. Both of these controllers will be connected to Virtual Device Server;
- A direct communication channel between the serial controller in `fv4` and the serial controller in `fv2`;

The overall count of emulated serial controllers for the subjects:

- `fv1` - 1 serial controller;
- `fv2` - 2 serial controllers;
- `fv3` - 2 serial controllers;
- `fv4` - 1 serial controller.

CHAPTER 17 *Virtual Shared Memory*

The LynxSecure® virtual shared memory feature allows the user to set up inter-subject communication flows using shared memory. These flows are similar to regular memory flows, except the memory and synthetic interrupts are managed by a virtual device.

Virtual shared memory devices can be used as a building block for creating inter-subject communication channels. For example, Virtual FIFO devices are supported by a Linux device driver provided with the LynxSecure distribution of Buildroot. This device driver detects all configured virtual shared memory devices and creates corresponding device nodes. These device nodes support either reading or writing (depending on the configured direction) using standard Linux® systems calls, enabling them to be used with standard Linux utilities such as `cat` or custom programs provided by the end-user.

Features

- LynxSecure supports up to 16 virtual shared memory devices per subject.
- Creates an inter-subject communication channel between subjects.

Limitations

- The virtual shared memory device requires a device driver. This release includes a driver for the Linux kernel in our Buildroot distribution that implements a FIFO on top of the virtual shared memory virtual device.

Autoconfig Syntax

Synopsis

```
autoconfig mkcv target-name \
--subject-fv1=virtfifo=DST_SUBJECT[:FIFO_NAME],...
--subject-fv1=virtshmem=DST_SUBJECT[:CONFIG-STRING:CONFIG-STRING-PEER[:SHMEM_NAME]],...
```

Subject Options

`virtfifo=`*`DST_SUBJECT`***`[:`***`FIFO_NAME]`*

Add a virtual shared memory device to a subject. This will create a shared memory region and interrupt. The virtual shared memory device will be configured in such a way that the lsk_fifo driver will be automatically loaded if available.

If FIFO_NAME is specified, it will be used as the name of the virtual device. If FIFO_NAME is not specified, the default name of `FIFO#` will be used.

`virtshmem=`*`DST_SUBJECT`***`[:CONFIG-STRING:CONFIG-STRING-PEER[:`***`SHMEM_NAME`***`]]`**

Add a virtual shared memory device to a subject. This will create a shared memory region and interrupt. The virtual shared memory device is not associated with any included device driver; the user must develop their own. If applicable the PCI_CLASS of this device is 0xFE, and the PCI_SUBCLASS is 0x00.

If specified, CONFIG-STRING and CONFIG-STRING-PEER will be set as the config string for the generated virtual device. CONFIG-STRING applies to the subject for which this option is specified. CONFIG-STRING-PEER applies to the DST_SUBJECT.

If SHMEM_NAME is specified, it will be used as the name of the virtual device. If SHMEM_NAME is not specified, the default name of `SHMEM#` will be used.

Sample Configuration

Let's assume the user wants to create bi-directional connectivity between two subjects

```
autoconfig mkcv target-name \
--subject-fv1=virtfifo=fv2 \
--subject-fv2=virtfifo=fv1:two_to_one \
```

The above command line will generate the following configuration:

- In subject fv1, a shared memory region with a write-only mode named `FIFO0` and a read-only hinted memory region named `two_to_one`.
- In subject fv2, a write-only hinted memory region named `two_to_one` and a read-only hinted memory region named `FIFO0`.

In this configuration, assuming that both fv1 and fv2 are running Linux images that include the `lsk_fifo` Buildroot package provided with the LynxSecure Buildroot distribution, both subjects will have FIFO files automatically created at /dev/lsk/fifo/FIFO0 and /dev/lsk/fifo/two_to_one. The permissions on these files will indicate the directionality of the FIFO.

NOTE: `lsk_fifo` driver on Arm® FV subject needs guest hypercall access to work. This can be enabled by using `perm=guesthypercall` subject specific option in the autoconfig command line.

CHAPTER 18 *Virtual USB (only x86 platform)*

The Virtual USB feature provides a rule-based mechanism for assigning individual physical USB devices to FV subjects.

This virtual device is only supported on the x86 platform.

LynxSecure® supports emulation for the following types of USB controllers:

- EHCI (USB 2.0);
- OHCI (USB 1.1);
- UHCI (USB 1.1).

The user can control aspects of the USB emulation:

- Select the type of the virtual USB controller;
- Set the maximum number of emulated devices;
- Select the devices which shall be emulated (filter by physical USB device path, device vendor, device product ID, device class, device subclass, etc.);
- Select the devices which shall not be emulated.

USB Emulation Rules

An emulation rule defines which of the physical USB devices should be assigned to which virtual USB controllers in which FV subject.

An emulation rule is a string of the following format: `owner[,filter]` where:

- `owner` is `OWNER-NAME[:VIRTUAL-USB-CONTROLLER-NAME];`
- `filter` is optional and has the following format: `[path=PATH], [class=CLASS[:SUBCLASS[:PROTOCOL[:CLASS-SPECIFIC-SETTINGS]]]],[id=VID[:PID]], [rev=REV],[sn=SERIAL],[hid=USAGE_PAGE[:USAGE_ID],[speed=MBITS],[kvm=KVM-SUBJECT-NAME].`

The rules are processed in the same order as they are added to the USB emulation rule list. If a rule is applicable to a specific device, any subsequent rules won't have any effect for that device at run-time. That means, there's no sense in having duplicate USB rules in the list.

There's no strict order of the attributes in the `device` specification.

If the `filter` attribute is not specified, all physical USB devices will be emulated for the subject specified by the `owner` attribute.

There's no limit for the length of the emulation rule list.

Table 18-1: Emulation Rules Specification

Field	Description
OWNER-NAME	Match the USB device owner by the subject name from the HCV or by the virtual USB controller alias. Forces the USB device to be assigned to the subject. If there is no virtual KVM, the user has to explicitly set the subject name (refer to the section “Features” (page 101)). Using `vds0` as a subject name will assign the matching device to the Virtual Device Server and exclude it from emulation. **`@static`** Instruct the emulation to assign the matching device to to the in-focus KVM subject at the time the device is connected; device assignment will not change if KVM focus is changed. Many targets have built-in USB devices such as a Bluetooth adapter, a web camera, a touchscreen, etc. Such devices cannot be physically detached from the target, so using `@static` will assign the device to the first KVM subject. The user can reassign the USB device to a different FV subject at run-time using the `usbadm` tool. Please refer to section “USB Administration Tool Syntax” (page 105) for more details. The user can also physically reattach the device (if it is possible) after the KVM focus is switched to a new subject. In this case, the recently attached device will be assigned to the current in-focus KVM subject. **`@follow`** Instruct the emulation to reassign the matching device to the new in-focus KVM subject every time the KVM focus is changed. This behavior removes the `@static` limitation of the device assignment to the current in-focus KVM subject at the time the device is attached (device assignment can only be changed by physically detaching the USB device or using the `usbadm` tool). For example, tablet PCs come with a touchscreen which is a built-in HID USB device. Emulating the touchscreen as a PS/2 mouse using a virtual KVM may not work on all tablets. On such targets, the only way to share the touchscreen device between several FV subjects is using USB emulation (OHCI or UHCI). For user convenience, the in-focus KVM subject should always have the touchscreen device in this case. That means, the touchscreen device should be dynamically reassigned the new in-focus KVM subject every time the KVM focus is changed, i.e. "follow the KVM focus". **`ALIAS`** Instruct the emulation to use virtual USB controller alias rather than the subject name for matching the USB device owner.
VIRTUAL-USB-CONTROLLER-NAME	Match the USB device owner by the virtual USB controller name from the HCV (must be available in *`OWNER-NAME`* namespace). This name could be obtained from the Autoconfig Tool output after SRP generation. Refer to section “Sample Configuration” (page 111). If `*` used as a virtual USB controller name emulation will automatically select the virtual USB controller based on the virtual USB bus speed (default behavior). The field is valid only if the *`OWNER-NAME`* field addresses the subject name, not the virtual USB controller alias.
path=*PATH*	Match the USB device by the device path on the target. This string is calculated based on the physical USB controller address followed by its relative path. This ID does not change with system restarts and always identifies the physical USB device path on the target [a].

Field	Description
	LynxSecure can match a part of the USB device path. Let's say if there are several USB devices whose USB device paths start with `usb-0000:00:1d.0`, then all USB devices could be matched by setting the *`path`* field of the emulation rule to `usb-0000:00:1d.0`.
`class=`*`CLASS`*`[:`*`SUBCLASS`*`[:`*`PROTOCOL`*`[:`*`CLASS-SPECIFIC-SETTINGS`*`]]]`	Match the USB device by class/subclass/protocol IDs [b], hex (e.g.: `03:01:02`) [a].
`hid=`*`USAGE_PAGE`*`[:`*`USAGE_ID`*`]`	Match the HID device against the HID Global Usage Page/ID, hex (e.g.: `01:06`, i.e. a keyboard).
`id=`*`VID`*`[:`*`PID`*`]`	Match the USB device by vendor/product IDs, hex (e.g.: `0403:6001`) [a].
`rev=`*`REV`*	Match the USB device by hardware revision number, hex (e.g.: `0331`) [a].
`sn=`*`SERIAL`*	Match the USB device by serial number (e.g.: `J9QO427I`) [a].
`speed=`*`MBITS`*	Match the USB device by the device speed in Mbits/s (e.g.: `480`, i.e. USB2.0 speed)[a].
`kvm=`*`KVM-SUBJECT-NAME`*	Match the USB device owner by the in-focus KVM subject name.

[a] `*` may be used if there's no need to match the particular device field.

[b] For class, subclass and protocol types refer to http://www.usb.org/developers/defined_class

Table 18-2: Examples

Rule	Description
`fv1:VIRTOHCI0,path=*,class=*,id=*,rev=*,sn=*` or `fv1:VIRTOHCI0`	Match any device and assign it to the FV subject `fv1` and virtual USB controller `VIRTOHCI0`.
`fv1:VIRTOHCI0,path=usb-0000:00:1d.0-1`	Match any device connected to USB device path `usb-0000:00:1d.0-1` and assign it to the FV subject `fv1` and virtual USB controller `VIRTOHCI0`.
`fv1:VIRTOHCI0,class=E0:01:01`	Match any device with class `E0`, subclass `01` and protocol `01` (i.e. any Bluetooth device) and assign it to the FV subject `fv1` and virtual USB controller `VIRTOHCI0`.
`fv1,path=usb-0000:00:1d.0-1`	Match any device connected to USB device path `usb-0000:00:1d.0-1` and assign it to the FV subject `fv1`. The virtual USB controller will be selected automatically based on the physical USB device speed.
`@static`	Match all USB devices and assign them to the in-focus KVM subject.
`@follow,class=03:01:02`	Match all USB mouse devices and re-assign them to the in-focus KVM subject once the KVM focus subject has changed.
`vds0,path=usb-0000:00:1d.0-1`	Do not emulate all devices connected to the USB device path `usb-0000:00:1d.0-1`.

Features

- By adding or removing emulation rules, the user makes a particular physical USB device appear or disappear in the guest OS. The rules may be configured statically by the Autoconfig Tool at SRP creation time, or dynamically in the VDS with the USB Administration Tool.
- LynxSecure tries to automatically assign the physical USB devices taking into account the physical USB device speed and the speed of the virtual USB controller. After testing the speed of all available virtual USB controllers in a FV subject, a best match is done based on the device speed: if the physical USB device speed is USB 2.0, then LynxSecure will try to use the virtual EHCI controller for the USB device emulation, if possible.

- If any FV subject in the configuration has the virtual VGA, keyboard and mouse devices, then the FV subject chosen for the automatic USB device assignment will be the first virtual KVM subject (the one that has the KVM focus initially). The user has to physically remove and reinsert the USB device in order to make it available in a different FV subject after KVM focus has changed. However, any explicit emulation rules specified at SRP creation time using Autoconfig Tool or specified at run-time using USB Administration Tool can override the default behavior.
- If any FV subject in the configuration has a virtual keyboard and mouse device, then all physical USB mice and keyboard devices will be automatically assigned to the VDS and won't be available to FV subjects. This behavior can be changed at run-time only.
- If all FV subjects in the configuration have no graphics emulation (i.e. no virtual VGA), e.g. have a directly assigned graphics controller or no graphics at all, LynxSecure cannot determine the default FV subject for automatic device assignment. In this case, the user has to explicitly specify the list of emulation rules, otherwise no devices will be emulated after the SRP is booted.
- If the USB emulation is enabled, the Autoconfig Tool attempts to assign as many physical USB controllers to the VDS as possible. Some physical USB controllers may not be assigned to the VDS, due to either an IRQ conflict with other devices or direct assignment to the other subject. Autoconfig Tool will issue a warning in this case. If VDS ends up with no flows to physical USB controllers, the physical USB device emulation will not be possible.

NOTE: By specifying the controller name `USB`*`x`* in the FV subject specification string of the Autoconfig Tool command line, the user assigns the physical USB controller (with all corresponding devices) to that FV subject. Devices attached to this controller cannot participate in the USB emulation.

- Virtual Device Server excludes the USB mass storage device from the USB emulation if the device is used as a virtual HDD/CD-ROM (BDE) for a FV subject.

Limitations

- LynxSecure supports up to 6 USB controllers of each type per a FV subject. All controllers of the same type will use the same emulated IRQ. MSI interrupts are not supported for all types of the virtual USB controllers.
- LynxSecure does not provide hotplug emulation of the virtual USB controllers. The user is allowed to specify the virtual PCI devices at SRP creation time only.
- Emulation does not support 64-bit addressing.
- The number of emulated USB devices per virtual USB controller is managed via the NDP setting. NDP stands for "Number of Downstream Ports" which is the number of root hub (RH) ports of the emulated controller. Currently, the USB emulation allows only one device to be emulated over one root hub port. This setting limits the maximum number of devices which the virtual USB controller can serve as follows: NDP equals to 'the number of RHs', which in turn, equals to 'the maximum number of devices per controller'.

Table 18-3: The maximum values of the NDP for different USB controller types

Type of the virtual USB controller	Maximum value for the NDP
EHCI	15
OHCI	15
UHCI	7

The theoretical maximum number of devices which can be emulated for one FV subject is 222 (which is 6 * (15 + 15 + 7)).

- The legacy interface mouse/keyboard is not supported for all types of virtual USB controllers.
- Port power switch emulation is not supported, all ports of all virtual USB controllers are emulated as powered on.
- Emulation of over-current detection is not supported.
- LynxSecure does not emulate pure virtual USB devices (such as USB mass storage using some other storage device as a backend, USB 1.1 or USB 2.0 hubs, etc.).
- LynxSecure does not emulate physical USB hubs.
- Some physical USB devices can be emulated using a particular virtual USB controller type only. For example, all physical USB 2.0 video camera devices cannot be emulated as connected using virtual OHCI or UHCI controller. Similarly, not all low- (USB 1.0) or full-speed (USB 1.1) devices can be emulated using a virtual EHCI controller (USB 2.0). For example, all HIDs (Human Interface Devices) such as USB mouse and keyboard cannot be emulated using a virtual EHCI controller.

  According to the EHCI specification, a USB 2.0 host controller can serve only high-speed (USB 2.0) devices. If a low- or full-speed device is connected to a USB 2.0 host controller, the EHCI specification defines the "port routing" logic and introduces a so-called "companion host controller". This approach assumes that the hardware implementation of the USB subsystem will include both USB 2.0 and USB 1.1 host controllers, e.g. both EHCI and OHCI. If an EHCI host controller receives the connect event on the USB port and the USB device speed on that port is less than the USB 2.0 device speed, the EHCI host controller hands off the device to one of its USB 1.1 companion controllers. Refer to Section 4.2 "Port Routing and Control" of the EHCI specification: http://www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/ehci-specification-for-usb.pdf

  However, despite the strict requirements of the EHCI specification, in some rare cases, the physical USB devices operating at the USB 1.1 speed could be emulated using a virtual EHCI controller. LynxSecure provides a way to visualize which virtual USB controllers can be used for the physical USB device emulation. Refer to section “List All USB Devices” (page 107).

  LynxSecure emulates port routing with the USB emulation rules and with making it possible to assign a virtual USB 1.1 controller to a FV subject at SRP creation time.
- Limited support for emulation of the physical USB devices which use isochronous (at equal time intervals) transfers.
- It is not possible to use the physical USB mass storage device as a boot device for FV subjects.
- VDS may require more memory if more virtual USB controllers were specified in the SRP.

Virtual EHCI Limitations

- EHCI v1.1 is not supported. See http://www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/ehci-v1-1-addendum.pdf
- EHCI Extended Capabilities are not supported. See section "5.1 EHCI Extended Capability: Pre-OS to OS Handoff Synchronization" of the EHCI specification.
- Companion controllers are not supported. The port routing logic is implemented by the means of USB emulation rules.
- Asynchronous Schedule Park Capability is not supported.
- Light controller reset is not supported.
- Debug Port ("Appendix C" of the EHCI specification) is not supported.
- Processing of FSTNs and siTDs is not supported.
- Split transactions are not supported.

List of Verified USB Devices

The following section provides the list of USB devices that have been successfully tested with LynxSecure in the current release.

Table 18-4: Tested USB devices

USB Device ID	USB Device Description	Virtual EHCI Compatibility	Virtual OHCI Compatibility	Virtual UHCI Compatibility
`8564:1000`	USB2.0 Transcend Mass Storage Device	PASS	PASS	PASS
`13fe:5500`	USB3.0 Patriot Mass Storage Device	PASS	PASS	PASS
`2001:1a02`	USB2.0 D-Link DUB-E100 rev.c1	PASS	FAIL	FAIL
`13b1:0041`	USB3.0 Linksys USB3GIGV1	PASS	FAIL	FAIL
`0424:7500`	USB2.0 USB Gigabit LAN [StarTech.com]	PASS	FAIL	FAIL
`0bda:8176`	USB2.0 802.11n WLAN Adapter	PASS	FAIL	FAIL
`05ca:1814`	USB2.0 Dell6410 Laptop Integrated Webcam	PASS	N/A	N/A
`1e4e:0100`	USB2.0 USB HD Webcam	PASS	N/A	N/A
`0e8d:1887`	USB2.0 Portable Super Multi Drive	PASS	N/A	N/A
`067b:2303`	USB-Serial Controller D (pl2303)	N/A	PASS	FAIL
`413c:2111`	Dell USB Wired Entry Keyboard	N/A	PASS	PASS
`046d:c077`	USB Optical Mouse	N/A	PASS	PASS
`0a81:0205`	PS2 to USB Converter	N/A	PASS	PASS
`04b5:0680`	HID Sensor Board DFUP2183ZA	N/A	PASS	PASS
`056a:0117`	MultiTouch Sensor	N/A	PASS	PASS

Autoconfig Syntax

Synopsis

```
autoconfig mksrp target-name \
  --subject-fv1= \
    virtehci[:ndp=NUMBER[:alias=ALIAS]],... \
    virtohci[:ndp=NUMBER[:alias=ALIAS]],... \
    virtuhci[:ndp=NUMBER[:alias=ALIAS]],... \
    ...
```

```
autoconfig mksrp target-name \
  --usbrule=RULE1 \
  --usbrule=RULE2 \
  ...
  --usbrule=RULEn \
  --virtusb-no-auto-rules \
```

Subject Options

virtehci[:ndp=*NUMBER***[:alias=***ALIAS***]],virtohci[:ndp=***NUMBER***[:alias=***ALIAS***]], virtuhci[:ndp=***NUMBER***[:alias=***ALIAS***]]**

Assigns a virtual EHCI/OHCI/UHCI controller with `ndp` root hubs to a FV subject.

`alias=`*`ALIAS`* sets the virtual EHCI/OHCI/UHCI controller alias which later can be used in the USB emulation rule. Alias must be unique, it must not match any of the subject names.

Global Options

--usbrule=*RULE*

Add a new USB emulation *`RULE`* to the end of the USB emulation rules list. Please refer to section "USB Administration Tool Syntax" (page 105) for a more detailed description.

Autoconfig Tool will fail to create the SRP if the same rule was specified more than once.

--virtusb-no-auto-rules

Suppress adding automatically generated USB emulation rules.

Without this option, Autoconfig Tool adds automatically generated rules to prevent USB device assignment conflicts:

- exclude USB mass storage devices from the USB emulation if they are used by BDE.

USB Administration Tool Syntax

The user can manage the USB emulation from VDS at run-time using the `usbadm` tool.

Add a USB Emulation Rule

Synopsis

```
usbadm rule insert POSITION RULE
```

Description

Add a new rule to the specified *`POSITION`* the USB emulation rule list, where *`POSITION`* can be `head` (insert at the start of the rule list), `tail` (insert at the end of the rule list) or a numeric position. The user is responsible for ensuring there are no duplicate rules.

Example

```
[root@vds0 ~]# usbadm rule insert head fv1:VIRTOHCI1,path=usb-0000:02.08.0-1
[root@vds0 ~]# usbadm rule insert tail fv1,path=usb-0000:00:1d.0-1.2.3
[root@vds0 ~]# usbadm rule insert tail @follow,class=03:01:02
[root@vds0 ~]# usbadm rule insert 1 vds0,class=03:01:01
```

Remove a USB Emulation Rule

Synopsis

```
usbadm rule remove POSITION
```

Description

Remove the rule at the specified *`POSITION`* from the USB emulation rule list, where *`POSITION`* can be `head` (remove from the start of the rule list), `tail` (remove from the end of the rule list) or a numeric position.

Example

```
[root@vds0 ~]# usbadm rule remove head
[root@vds0 ~]# usbadm rule remove tail
[root@vds0 ~]# usbadm rule remove 2
```

List All USB Emulation Rules

Synopsis

```
usbadm show rules [--daemon-conf|-d] [--canonical|-c] [--quiet|-q]
```

Description

List all emulation rules.

Table 18-5: Output formats

Option	Description
--daemon-conf|-d	Print rules in the USB daemon configuration file format.
--canonical|-c	Print rules in the canonical format.
--quiet|-q	Suppress verbose output.

Example

```
[root@vds0 ~]# usbadm show rules
# Output format:
#    {position} owner[,filter] - implicit rule (cannot be moved/deleted)
#    <position> owner[,filter] - user-defined rule
# where:
#    position - Rule position in the list
#    owner    - OWNER-NAME[:VIRTUAL-USB-CONTROLLER-NAME]
#    filter   - [path=PATH],[class=CLASS[:SUBCLASS[:PROTOCOL[:CLASS-SPECIFIC-SETTINGS]]]],
#               [id=VID[:PID]],[hwrev=HWREV],[sn=SERIAL],[hid=[USAGE_PAGE[:USAGE_ID]]],
#               [speed=MBITS],[kvm=KVM-SUBJECT-NAME]
#
# Run `help' command to see more details.

{0} vds0,class=09
<1> @follow,hid=0d:02
```

```
<2> @follow,hid=0d:03
<3> @follow,hid=0d:04
<4> @follow,hid=0d:05
<5> @follow,hid=0d:20
<6> @follow,id=056a:00ec
<7> vds0,hid=01:02
<8> vds0,hid=01:06
<9> vds0,class=03:01:01
<10> vds0,class=03:01:02
<11> @static
```

Debugging Level

Synopsis

```
usbadm debug DEBUG-LEVEL
```

Description

Change the USB emulation debugging level. All logs are available in VDS: `/var/log/lsk/daemons/usbd.log`.

Table 18-6: The USB emulation debugging levels

The USB emulation debugging level	Description
0	Log only errors (default).
1	Add more messages to the logs.
2	The maximum level of verbosity.

Example

```
[root@vds0 ~]# usbadm debug 2
```

Display the Debugging Level

Synopsis

```
usbadm show debug
```

Description

Show the debugging level of the USB emulation.

List All USB Devices

Synopsis

```
usb show devices [--machine-readable|-m|--pretty|-p] [--descriptors|-d] [--expand-owner|-e]
```

Description

List all USB devices and their assignments.

Table 18-7: Output formats

Option	Description
--machine-readable|-m	Print all information in machine-readable format for use by 3rd party scripts.
--pretty|-p	Print all information in human-readable manner.
--descriptors|-d	Dump all device descriptors for each USB device.
--expand-owner|-e	Print all information about the owner, including the virtual USB controller alias.

By default, the following information is displayed:

Table 18-8: Default output format

Field	Description
bus.device	The logical address of the USB device in the VDS.
path	The USB device path.
subject[:hc]	The USB device assignment: subject name and virtual USB controller name respectively.
vid:did	The USB device vendor/product ID in hex.
description	A brief physical USB device description.

Example

```
[root@vds0 ~]# usbadm show devices
1.1 usb-0000:00:1a.0 vds0 1d6b:0002 EHCI Host Controller
1.2 usb-0000:00:1a.0-1 vds0 8087:0020 Hub 8087:0020
1.3 usb-0000:00:1a.0-1.1 vds0 413c:2513 Hub 413c:2513
1.6 usb-0000:00:1a.0-1.1.2 vds0 045e:0040 Microsoft 3-Button Mouse with IntelliEye(TM)
1.7 usb-0000:00:1a.0-1.1.3 vds0 413c:2111 Dell USB Wired Entry Keyboard
1.4 usb-0000:00:1a.0-1.2 vds0 413c:2513 Hub 413c:2513
1.5 usb-0000:00:1a.0-1.4 fv1 05ca:1814 Laptop Integrated Webcam 3M
2.1 usb-0000:00:1d.0 vds0 1d6b:0002 EHCI Host Controller
2.2 usb-0000:00:1d.0-1 vds0 8087:0020 Hub 8087:0020
2.3 usb-0000:00:1d.0-1.3 vds0 1a40:0201 USB 2.0 Hub [MTT]
2.6 usb-0000:00:1d.0-1.3.3 fv1 0951:1601 DataTraveler II+
2.7 usb-0000:00:1d.0-1.3.4 fv1 13fe:5500 Patriot Memory
2.8 usb-0000:00:1d.0-1.3.6 fv1 0e8d:1887 Portable Super Multi Drive
2.9 usb-0000:00:1d.0-1.3.7 vds0 1a40:0101 USB 2.0 Hub
2.10 usb-0000:00:1d.0-1.3.7.1 fv1 8564:1000 Mass Storage Device
2.11 usb-0000:00:1d.0-1.3.7.4 fv1 0c45:6341 USB 2.0 Camera
2.4 usb-0000:00:1d.0-1.7 fv1 413c:8187 DW375 Bluetooth Module
2.5 usb-0000:00:1d.0-1.8 fv1 0a5c:5801 USB 0a5c:5801
```

Display the USB Emulation Status

Synopsis

```
usbadm show status
```

Description

List all virtual USB controllers and their assignments.

Table 18-9: Output format

Field	Description
subject:hc	The name of the FV subject owning the virtual USB controller and the virtual USB controller name.
alias	The virtual USB host controller alias.
tag	The virtual USB controller identifier; currently, it is the PCI Bus/Device/Function ID of the corresponding virtual device.
numberOfEmulatedDevices	The number of the USB devices currently being emulated over the virtual USB controller.
maxNumberOfEmulatedDevices	The maximum number of the USB devices which can be emulated over the virtual USB controller.
type	The virtual USB controller type.
status	The virtual USB controller status.

Example

```
[root@vds0 ~]# usbadm show status
fv2:VIRTOHCI1 - 00050 0 15 12 OK
fv2:VIRTEHCI1 - 00060 0 15 480 OK
fv1:VIRTOHCI0 myalias 00050 2 15 12 OK
fv1:VIRTEHCI0 - 00060 6 15 480 OK
```

Show Help

Synopsis

```
usbadm [-h|--help]
```

Description

Display a brief help.

Sample Configuration

Let's assume we've got a target with the attached physical USB devices (the list of the physical USB devices for a particular target can be obtained with the `autoconfig show ...` command, refer to section “Working with Target System Information” (page 15)):

We shall demonstrate how to manage the USB emulation for a two FV subjects configuration.

```
autoconfig mksrp target \
--subject-fv0=virthd=bde1,virte1000=NAT,virtkbdmouse,virtvga,virtehci,virtohci \
--subject-fv1=virthd=bde2,virte1000=NAT,virtkbdmouse,virtvga,virtehci,virtohci
```

This is the simplest example where LynxSecure will try to automatically assign all physical USB devices.

After booting up, `fv0` will claim all the physical USB devices since this is the first FV subject which has the virtual KVM focus.

However, a physical USB device could be 'handed off' to another FV subject. While `fv0` has the virtual KVM focus, the user can physically unplug the device (if possible), then switch to `fv1` using the **Ctrl-Shift-F2** combination, and then plug in the USB device into the same USB device path. The device will become available to the `fv1` subject.

Once the system is booted, the user can log in to the VDS over SSH and use USB Administration Tool to manage the emulation rules (refer to section “List All USB Devices” (page 107), section “Add a USB Emulation Rule” (page 105)):

```
[root@vds0 ~]# usbadm show status
fv0:VIRTEHCI0 - 00040 0 15 USB2.0 OK
fv0:VIRTOHCI0 - 00030 0 15 USB1.1 OK
fv1:VIRTEHCI1 - 00040 0 15 USB2.0 OK
fv1:VIRTOHCI1 - 00030 0 15 USB1.1 OK
[root@vds0 ~]# usbadm rule insert head fv0,path=usb-0000:00:1a.0-1.1.1.2.4
```

This will assign the physical USB device in the USB device path `usb-0000:00:1a.0-1.1.1.2.4` to the FV subject `fv0`. The virtual USB controller with the most suitable speed will be selected automatically.

```
[root@vds0 ~]# usbadm rule insert head fv1:VIRTOHCI1,path=usb-0000:00:1d.0-1.7
```

This will assign the physical USB device in the USB device path `usb-0000:00:1d.0-1.7` to the virtual OHCI controller #1 in the FV subject `fv1`.

```
[root@vds0 ~]# usbadm rule insert head fv1,path=usb-0000:00:1a.0-1.1.3
```

This will assign the physical USB device in all USB device paths starting with `usb-0000:00:1a.0-1.1.3` to the FV subject `fv1`.

```
[root@vds0 ~]# usbadm rule insert head vds0,path=usb-0000:00:1a.0-1.1.1.3.2
```

This will stop emulating the device in USB device path `usb-0000:00:1a.0-1.1.1.3.2` for `fv0`. The device will be excluded from the emulation altogether. Any subsequent emulation rules will be ignored for that USB device.

In order to have the same emulation rules at SRP boot time, the user has to specify `--usbrule=...` options on the Autoconfig Tool command line and re-generate the SRP:

```
autoconfig mksrp target \
  --subject-fv0=virthd=bde1,virtkbdmouse,virtvga,virtehci,virtohci \
  --subject-fv1=virthd=bde2,virtkbdmouse,virtvga,virtehci,virtohci \
  --virtusb-no-auto-rules \
  --usbrule="vds0,class=03:01:02" \
  --usbrule="vds0,class=03:01:01" \
  --usbrule="fv0,path=usb-0000:00:1a.0-1.1.1.2.4" \
  --usbrule="fv1,path=usb-0000:00:1a.0-1.1.3" \
  --usbrule="fv1:VIRTOHCI1,path=usb-0000:00:1d.0-1.7" \
  --usbrule="vds0,path=usb-0000:00:1a.0-1.1.1.3.2" \
  --usbrule="@static"
```

One of the problems here is how to identify the virtual USB controller name.

- One way to do it is using the Autoconfig Tool command line with the `-g` option. The first run will display the device map:

```
Subject "fv0", type FULLVIRT, memory 1.00GiB, 1 vCPU(s)
        initparams=BOOTMENUTIMEOUT:0 CD:1 DISK:2 BEV:3 KDI:4
Name             Device ID  Address  IRQ CanSep                    Flag Description
====             =========  ======= ==== ======                    ==== ===========
VIRTAHCI0        8086:2922  xx:xx.x auto                           AHCI Virtual Block Device: ...
VIRTEHCI0        1cab:4d0b  xx:xx.x auto                         ndp=15 Virtual USB Controller: ...
VIRTNET_1        8086:100e  xx:xx.x auto              02:00:37:29:4b:01 Virtual Ethernet Controller: ...
VIRTOHCI0        1cab:4d05  xx:xx.x auto                         ndp=15 Virtual USB Controller: ...
VIRTVGA0         1cab:4d08  xx:xx.x                                     Virtual VGA
LOGGER_FV0                                                              Logger
MONITOR_FV0                                                             Subject Monitor
VIRTKBDMOUSE0    0000:0303          1/12     No                         Virtual Keyboard/Mouse

Subject "fv1", type FULLVIRT, memory 1.00GiB, 1 vCPU(s)
        initparams=BOOTMENUTIMEOUT:0 CD:1 DISK:2 BEV:3 KDI:4
Name             Device ID  Address  IRQ CanSep                    Flag Description
====             =========  ======= ==== ======                    ==== ===========
VIRTAHCI1        8086:2922  xx:xx.x auto                           AHCI Virtual Block Device: ...
VIRTEHCI1        1cab:4d0b  xx:xx.x auto                         ndp=15 Virtual USB Controller: ...
VIRTNET_2        8086:100e  xx:xx.x auto              02:00:37:29:4b:02 Virtual Ethernet Controller: ...
VIRTOHCI1        1cab:4d05  xx:xx.x auto                         ndp=15 Virtual USB Controller: ...
VIRTVGA1         1cab:4d08  xx:xx.x                                     Virtual VGA
LOGGER_FV1                                                              Logger
MONITOR_FV1                                                             Subject Monitor
VIRTKBDMOUSE1    0000:0303          1/12     No                         Virtual Keyboard/Mouse
```

Now the names of the virtual USB controllers are known and could be used in `--usbrule=...`.

CAUTION! Modifying the list of FV subjects in the Autoconfig Tool command line or the list of the virtual USB controllers assigned to any FV subject will change the names of the virtual USB controllers.

- Another way is defining an alias for the virtual USB controller and using it in the USB rule:

```
autoconfig mkcv target \
  --subject-fv0=virthd=bde1,virtkbdmouse,virtvga,virtehci,virtohci \
  --subject-fv1=virthd=bde2,virtkbdmouse,virtvga,virtehci,virtohci:alias=MYALIAS \
  --virtusb-no-auto-rules \
  --usbrule="vds0,class=03:01:02" \
  --usbrule="vds0,class=03:01:01" \
  --usbrule="fv0,path=usb-0000:00:1a.0-1.1.1.2.4" \
  --usbrule="fv1,path=usb-0000:00:1a.0-1.1.3" \
  --usbrule="MYALIAS,path=usb-0000:00:1d.0-1.7" \
  --usbrule="vds0,path=usb-0000:00:1a.0-1.1.1.3.2" \
  --usbrule="@static"
```

CHAPTER 19 *LSA.store Full Disk Encryption (only x86 platform)*

The LSA.store feature allows the user to encrypt virtual block devices for a FV subject. LSA.store is an optional component that is licensed separately from LynxSecure®. If you are interested in this feature but do not have a license, please contact your Lynx Software Technologies sales representative.

LSA.store is only supported on the x86 platform.

LSA.store provides transparent full-disk encryption to fully virtualized subjects. This encryption protects the confidentiality and (optionally) the integrity of virtual disks by implementing industry standard (IEEE P1619 and IEEE P1619.1) cryptographic techniques while leveraging the architectural tools provided by LynxSecure to minimize the opportunity for adversaries to compromise or bypass the encryption.

When using LSA.store with LynxSecure, a subset of the virtual disk controllers and types emulated by Block Device Emulation are available. Specifically, LSA.store supports the Intel® 82801IR ICH9 Digital Office (ICH9R) Controller with virtual hard disks. Legacy IDE and optical media emulation are not supported.

This section provides a detailed description of LSA.store, Autoconfig Tool syntax, and basic examples. This section builds on Chapter 13, "*Virtual Block Device (only x86 platform)*" (page 81). The basic features and limitations of LSA.store are inherited from Block Device Emulation; additional features, limitations, and examples specific to the use of LSA.store are provided here.

Features

Cryptography

For each virtual disk provided to a subject, the user can specify the use of unauthenticated encryption, authenticated encryption, or no encryption. Unauthenticated encryption protects the confidentiality (but not the integrity) of a virtual disk. Specifically, this means that an attacker with access to the Block Device Emulation backend device used by the subject can not read the data on the device, but can corrupt the data seen by a subject. Authenticated encryption builds on the protection provided by unauthenticated encryption by preventing any modified data from reaching the subject. Modifications to the data will result in zero-filled sectors being returned. When a virtual block device is configured with no encryption, the raw data provided by the back-end device is provided to the subject without modification by LSA.store.

Key Management

Each encrypted virtual disk has an associated set of key material. This key material must be made available to LSA.store at run time. Key material can be embedded in the SRP or directly in the disk image. Included with LSA.store is tooling for creating in-SRP and in-disk configurations. This tool manages the creation, encryption, and packaging of key material in both in-SRP and in-disk configurations. Additionally this tool provides a mechanism for importing and exporting key material.

Host Based Disk Cryptography

To ease deployment of solutions based on LSA.store, also included is a tool (`lsastore crypt-disk`) for encrypting and decrypting disk images on an arbitrary host machine. This tool utilizes the same cryptographic library as LSA.store and produces binary compatible images.

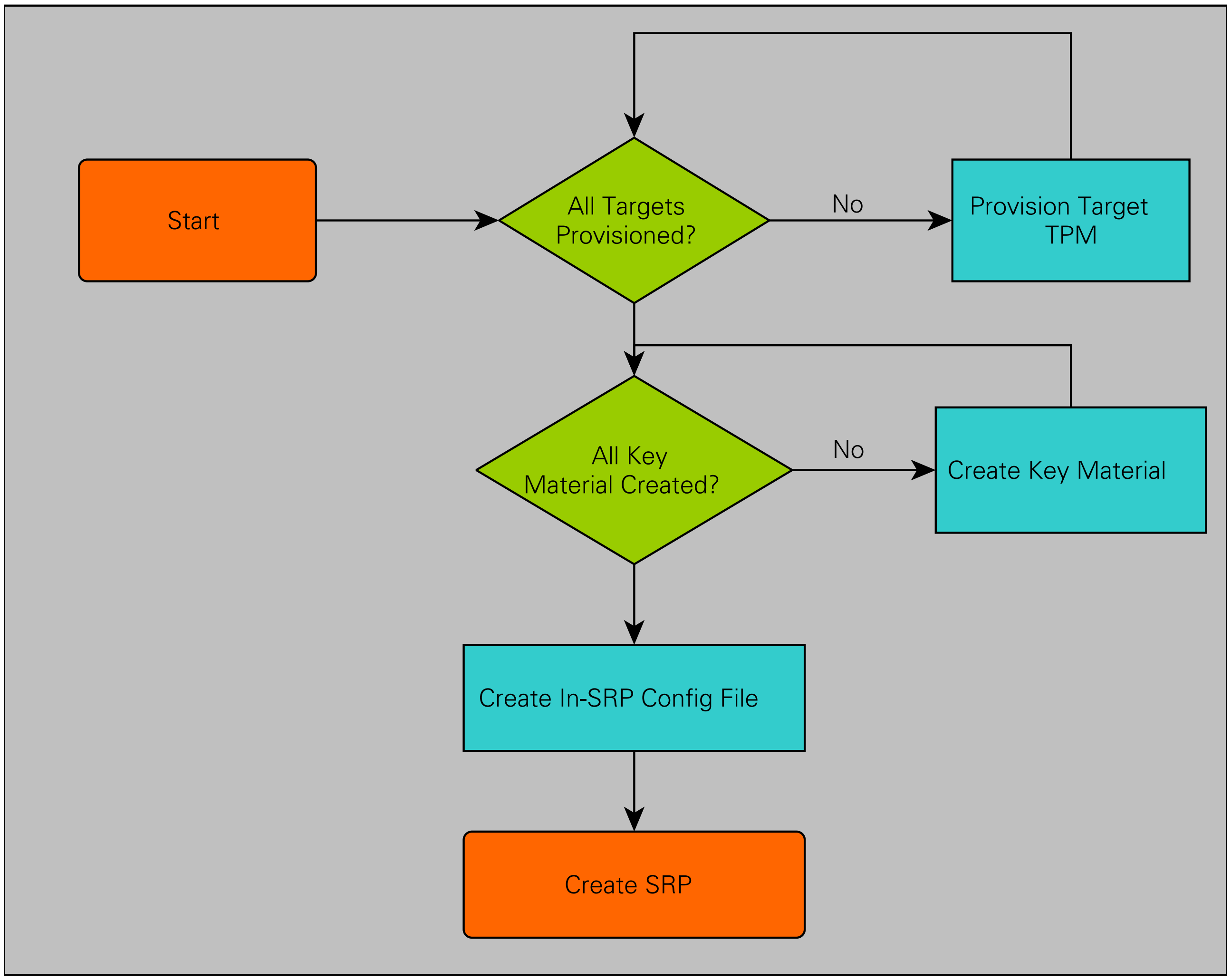
Workflow Overview

Outlined in this section are the basic steps required to configure, deploy, and manage LSA.store. It is assumed that the reader already has familiarity with the base operation of LynxSecure and that the target to be utilized in conjunction with LSA.store has already be configured for use with LynxSecure.

Creating In-SRP Configurations

Creating an in-SRP configuration requires that all key material is established before the SRP is created. This involves first provisioning the TPM on each target if needed, creating key material for each disk, creating an in-srp configuration file, and finally creating the SRP.

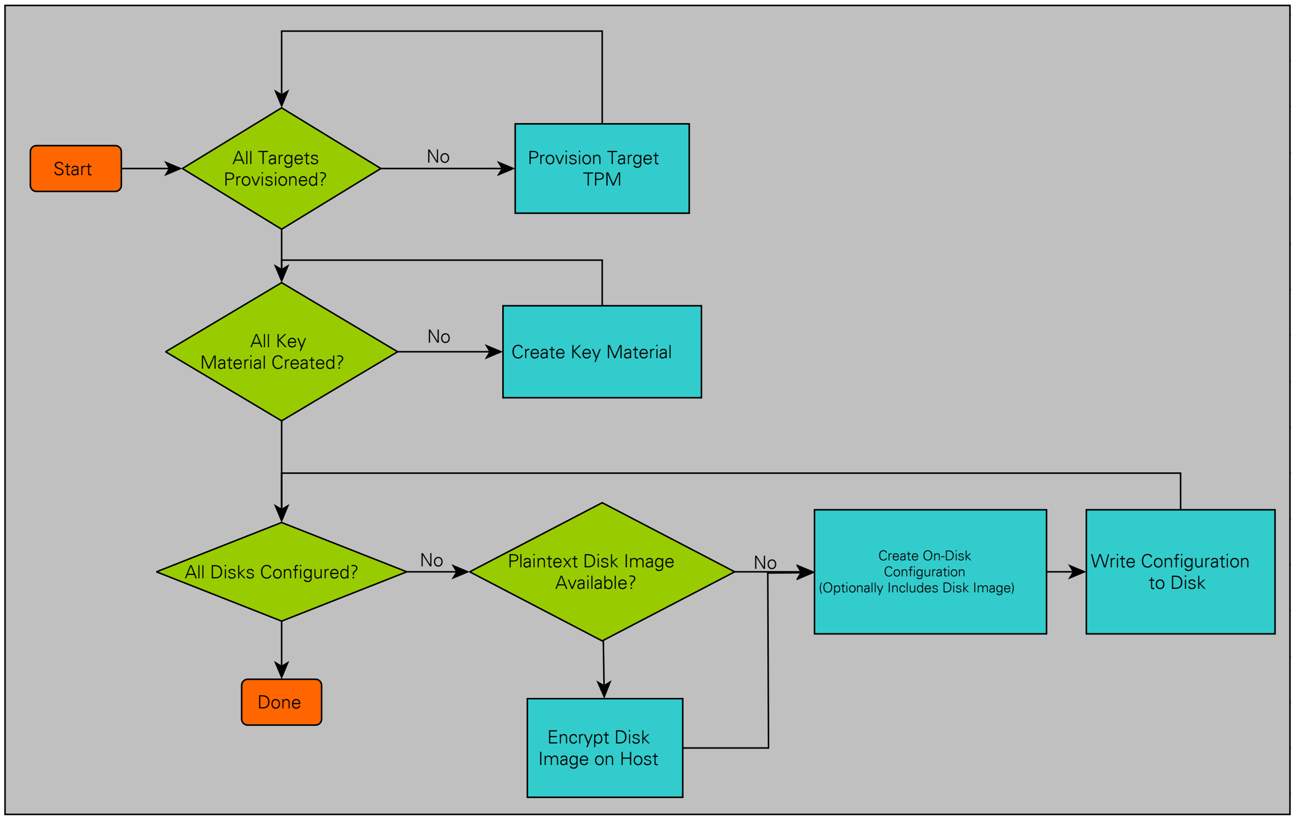
Figure 19-1: In-SRP Workflow



Creating In-Disk Configurations

Creating an in-disk configuration allows for the SRP to be generated before the key material is created or the targets TPMs are provisioned. In this workflow you would first create the SRP, then provision the TPMs on the targets, create the key material, and write the key material directly to the target disks.

Figure 19-2: In-Disk Workflow


Start
All Targets Provisioned?
No
Provision Target TPM
All Key Material Created?
No
Create Key Material
All Disks Configured?
No
Plaintext Disk Image Available?
No
Create On-Disk Configuration (Optionally Includes Disk Image)
Write Configuration to Disk
Done
Encrypt Disk Image on Host

Configuring the Target's TPM 1.2

In-SRP key material can optionally be protected by a target's TPM 1.2. In-disk key material requires a TPM 1.2. To use a TPM 1.2 with an LSA.store deployment it must first be provisioned. This process comprises resetting the TPM to the unowned state and generating sealed key material. To reset the TPM to the unowned state please consult your hardware manuals. Once the TPM 1.2 is in the unowned state, LSA.store includes tools for provisioning the TPM 1.2. Provisioning the TPM 1.2 creates an EK, sets the well known password, and generates PGP keys that are sealed to the particular TPM 1.2. These keys are usable to encrypt both in-SRP and in-disk key material.

Creating Key Material for Virtual Disks

Key material is generated using tools provided as part of LSA.store. These keys are optionally encrypted for a given target. Key material can be packaged as part of the SRP, or as part of a disk image. When packaged as part of a disk image, the sealed key material for a given target is directly included in the disk image. This relies on the TPM 1.2 to provide confidentiality. It is also possible to create a configuration that includes a pre-created encrypted disk image. These can be directly written to the backing storage device.

Creating SRPs that enable LSA.store

The command line used to generate the HCV for a given configuration must be modified to include the resources necessary for LSA.store. These include LSA.store itself, and any TPM resources that are needed for ensuring the confidentiality of the key material.

Creating Encrypted Disk Images

Disk images can be created in one of two ways. First, if no plaintext disk image is available the disk can be configured at runtime. For example, if the goal is to install Windows®, the subject could be provisioned with a install drive provided via virtual usb, and an encrypted disk provided via LSA.store. If a plaintext disk image is already available, LSA.store includes tools for encrypting these disk images on a host machine.

Deploying Encrypted Disk Images

The disk image format depends on the configuration of LSA.store. If an in-SRP configuration is deployed, then encrypted disk images can be directly written to the backing device. If an in-disk configuration is desired, then the tooling associated with LSA.store is capable of combining the in-disk encryption keys with the supplied disk image. Note that in-disk configurations require extra storage for storing key data. Be careful not to over-provision your disks.

Managing Keys

Once an encrypted disk image has been created, it is often useful to allow multiple targets to use this image. The tooling included with LSA.store allows for a single set of key material to be encrypted for multiple targets. With in-SRP configurations, this happens before the SRP is generated. Each target would have its own SRP with its own key material. For in-disk configurations this happens independently of SRP creation.

Importing and Exporting Key Data

When provisioning the TPM 1.2 on a target, key data is generated. If this key data is generated locally on the target, it must be imported to the host machine before disk configurations can be made to utilize it. Further if multiple host machines are in use, key material can be exported from one host to another. The tooling included with LSA.store supports these use cases.

Configuring the Target's TPM 1.2

The TPM 1.2 for a given target can be configured in one of two ways: Remotely from a host machine, or directly on the target using the Hardware Discovery Linux®. It is assumed that the target is already running Hardware Discovery Linux at this stage.

Provisioning the TPM 1.2 from the host

If the target machine is available via the network, then it can be provisioned from the host machine. `lsastore target-provision` is used to achieve this. For example, if a target named test-target1 is available at 10.20.31.123:

```
lsastore target-provision 10.20.31.123 test-target1
```

This command accesses the target via SSH, takes ownership of the TPM, and generates sealed key material specific for this target. This key material is automatically imported to the host.

Provisioning the TPM 1.2 from the target

If the target is not available via the network, then the TPM must be provisioned by running commands locally from the Hardware Discovery Linux shell. For example, if a target name test-target1 is running Hardware Discovery Linux, and there is a usb drive plugged in mounted at /mnt, then the following command will provision the TPM:

```
lsastore-provision test-target1 /mnt/
```

This command will take ownership of the TPM, and create and write an archive containing key data to /mnt. This key data must next be imported to the host machine:

```
lsastore import /path/to/testtarget1.tar
```

Note: As a pre-condition, the TPM 1.2 must be reset to an unowned state before these instructions are executed. Please consult your hardware manuals for instructions on how to reset the state of your TPM 1.2.

Creating Key Material for Virtual Disks

Creating configurations for a given virtual disk is achieved using `lsastore disk-config-create`. This command creates and encrypts a given configuration for its associated targets. For example, to create a disk configuration that enables AES-XTS mode for targets named test-target1 and test-target2, named diskConfig1:

```
lsastore disk-config-create diskConfig1 --aes test-target1 test-target2
```

And to create a similar config but in bypass mode:

```
lsastore disk-config-create diskConfig1 --bypass test-target1 test-target2
```

After running one of these commands a new config is created, and encrypted using the host machine's key. Any number of targets may be associated with a given disk configuration. Once a disk config is created additional targets can be added to or removed from the given config. For example to add test-target3 to test-disk config:

```
lsastore disk-config-modify --append diskConfig1 test-target3
```

Once all disk configs are created and all desired targets are added to those configurations, the in-disk or in-SRP configuration can be generated. For example, to generate an encrypted in-SRP configuration for the target test-target1 including keys for disk test-disk1, test-disk2, and test-disk3:

```
lsastore builtin-config -d test-disk1 0 -d test-disk2 1 -d test-disk3 3 \
    -e test-target1 lsastore-test-target1.conf
```

This command would include the configurations for test-disk 1, 2, and 3, encrypt them for test-target1, and write the encrypted configuration to lsastore-test-target1.conf. This configuration file is suitable for directly including in an SRP.

To create an in-disk config for test-disk1, and package it with an existing encrypted image named test-disk1.bin:

```
lsastore diskpack-create -d test-disk1.bin test-disk1 test-disk1.pack
```

This command assembles all of the required key material together with the pre-encrypted disk image in a format suitable for directly writing to the backing storage device. This step must be repeated for each disk image.

Autoconfig Syntax

Synopsis

```
autoconfig mksrp target-name \
--subject-vds=tpmsecret[=FILENAME] \
--subject-fvSUBJECT_NAME= \
virthd=[DEVICE-NODE|GPT-LABEL|NFS-URL|FILE],... \
LSA.store[=FILENAME],... \
...
```

VDS Options

If TPM 1.2 based confidentiality is desired, then TPM and ACPI resources must be assigned to VDS. The option `tpmsecret` does this. Optionally, a sealed encryption key can be specified for inclusion in the SRP. If this key is present, this allows in-SRP based disk configurations to be encrypted. This key is not required for in-disk configurations.

Subject options

virthd=*DEVICE-NODE***|***GPT-LABEL***|***NFS-URL***|***FILE***]**

Specify the virtual block device partition. LSA.store will handle this device according to the specification provided in the LSA.store configuration file.

LSA.store*[=FILENAME]*

Specify the LSA.store configuration file for use by this subject. Configuration files include the encryption mode for each disk as well as any associated cryptographic keys and associated cryptographic variables. If no configuration is specified, then it is assumed that LSA.store will operate with in-disk configurations. In this case, no configuration will be included in the SRP.

Each virthd has an associated zero-based index determined by the order the device was specified on the autoconfig command line. These indices are required when generating an in-SRP configuration to associate the key material with a given disk.

Note that by default the configuration file is included directly in the SRP. Attackers with access to the SRP can read the contents of the configuration file. LSA.store has support for encrypting its configuration file by leveraging a TPM in the scenario where the SRP must also be protected. On disk configurations are always protected by the TPM.

If this is not sufficient to meet your system security goals, please contact Lynx Software Technologies to discuss your options for alternative key management options.

Creating Encrypted Disk Images on the Host

If a plaintext disk image is available, then it can be encrypted for a give disk configuration from the host machine. For example, to encrypt plaintext-disk1.bin for the config test-disk1:

```
lsastore disk-crypt test-disk1 plaintext-disk1.bin encrypted-disk1.bin
```

This command reads the keys from the test-disk1 configuration, encrypts plaintext-disk1.bin using these keys, and writes it to encrypted-disk1.bin.

Importing/Exporting data from a Host System

When multiple host systems are used to configure targets including LSA.store, it will some times be necessary to move key material between host systems. To facilitate this, LSA.store's tooling includes import and export functionality. Since all key material is stored encrypted on the hosts disks, it is first necessary to obtain the public key for a give host. For example, if you wish to export you data from Host1 to Host2, you must first obtain the public key of Host2 by running the following command on Host2:

```
lsastore kmw-export-pubkey output_directory/
```

This command will create a file named Host2_XXXXXXXXXXXXXXXX.pub, where the X's are replaced with the fingerprint of Host2. This key should be copied to Host1 and imported:

```
lsastore kmw-import-pubkey Host2_XXXXXXXXXXXXXXXX.pub
```

Having this key in the keyring of Host1 allows us to export the key material from Host1 in a form encrypted for Host2:

```
lsastore export XXXXXXXXXXXXXXXX Host1.tar
```

The above command will encrypt all of the key material for Host2, and export it to a file name Host1.tar. This file can then be copied to Host2 and imported:

```
lsastore import Host1.tar
```

Setting up the Key Management Tools in a Standalone Environment

All of the tools for managing LSA.store keys are included in the default CDK. If it is desired to manage keys in a non-CDK environment the tooling is provided as a separate RPM. This RPM is compatible with any version of CentOS™ 7 starting with CentOS 7.3. Once installed, the tools are available in `/opt/lynxsecure/6.3.0-rev16326/lsa-store-kmw/`. To enable the use of these tools first execute the following from a bash prompt:

```
source /opt/lynxsecure/6.3.0-rev16326/lsa-store-kmw/activate
```

Once this setup is executed, `lsastore` is available on the path.

Limitations

- The use of legacy IDE controllers is not supported.
- The use of optical media (e.g., DVD images) is not supported.
- In-disk configuration encryption requires a TPM 1.2. Other versions of TPM, and other hardware security modules are not supported.

APPENDIX A *Example Configurations*

This section includes different LynxSecure® 32-bit and 64-bit configurations that will help the user to get started. For the list of supported Guest OSes refer to the section "Verified Guest OSes" in *LynxSecure 6.3.0 Release Notes*.

Pre-requisite:

Verify that the following tasks are completed prior to LynxSecure SRP creation:

- LynxSecure Cross Development environment location: `/opt/lynxsecure/6.3.0-rev16326/x86_64`

  The environment may be cloned as described in section "Multi-User Environment and Multiple Projects" (page 5).
- Install all hardware components prior to setting up the system
- Configure the target as described in the Chapter 3, "*Setting up the Target System*" (page 11)
- Collect the Hardware information for the target by following the procedures in the section "Working with Target System Information" (page 15)
- If necessary, create virtual disk partitions, and if the Guest OS has not been installed yet, refer to the Chapter 6, "*Installing Guest Operating Systems*" (page 41)

Para-Virtualized Linux Configuration

This section provides example configurations for Para-Virtualized (PV) Linux® guests.

One PV Linux with Direct Device Assignment

Directly assigned VGA0, graphics, mouse, keyboard, Ethernet to PV Linux with default memory (256MB) and 1 CPU.

```
host$ autoconfig mksrp target \
    --subject-pvlinux1=kernel=pvlinux-vmlinux-x86_64.bin,\
ramdisk=pvlinux-ramdisk-x86_64.img,VGA0,GRAPHICS*,MOUSE0,KEYBOARD0,SERIAL0,NET0 \
    --output=/var/lib/tftpboot
```

The PV Linux comes up on the monitor and serial.

NOTE: LynxSecure only includes prebuilt VDS images; it is expected that the customers will build the PV Linux kernel and ramdisk configured as needed by their configuration. In this appendix, the names `pvlinux-vmlinux-x86_64.bin` and `pvlinux-ramdisk-x86_64.img` refer to thus rebuilt kernel and ramdisk. For trying out these configurations, it is possible to substitute VDS images instead: `vds-vmlinux-x86_64.bin` and `vds-ramdisk-x86_64.img`.

One PV Linux with Direct Device Assignment and Virtual Network

PV Linux with directly assigned VGA0, graphics, mouse, keyboard, Ethernet and virtual network in bridge mode, with default memory (256MB) and 1 CPU.

```
host$ autoconfig mksrp target \
    --subject-pvlinux1=virtnet,kernel=pvlinux-vmlinux-x86_64.bin,\
ramdisk=pvlinux-ramdisk-x86_64.img,VGA0,GRAPHICS*,MOUSE0,KEYBOARD0,SERIAL0 \
    --output=/var/lib/tftpboot
```

Above and below commands will create the same configuration.

```
host$ autoconfig mksrp target \
    --subject-pvlinux1=kernel=pvlinux-vmlinux-x86_64.bin,\
ramdisk=pvlinux-ramdisk-x86_64.img,VGA0,GRAPHICS*,MOUSE0,KEYBOARD0,SERIAL0,virtnet \
    --output=/var/lib/tftpboot
```

The PV Linux network interface will get the IP address assigned dynamically from the DHCP server that the target is connected to.

PV Linux with directly assigned VGA0, graphics, mouse, keyboard, Ethernet and virtual network in NAT mode, with default memory (256MB) and 1 CPU.

```
host$ autoconfig mksrp target \
    --subject-pvlinux1=kernel=pvlinux-vmlinux-x86_64.bin,\
ramdisk=pvlinux-ramdisk-x86_64.img,VGA0,GRAPHICS*,MOUSE0,KEYBOARD0,SERIAL0,virtnet=NAT \
    --output=/var/lib/tftpboot
```

The PV Linux network interface will get the 10.1.0.2 IP address assigned dynamically from VDS.

One PV Linux with Direct Device Assignment and Virtual UART

PV Linux with directly assigned VGA0, graphics, mouse, and keyboard, virtual UART connection to VDS, with default memory (256MB) and 1 CPU.

```
host$ autoconfig mksrp target \
    --subject-pvlinux1=kernel=pvlinux-vmlinux-x86_64.bin,\
ramdisk=pvlinux-ramdisk-x86_64.img,VGA0,GRAPHICS*,MOUSE0,KEYBOARD0,SERIAL0,virtuart \
    --output=/var/lib/tftpboot
```

Virtual serial connection between PV Linux and VDS is configured.

Multiple PV Linux Subjects with Virtual Devices

Multiple PV Linux subjects. One subject has directly assigned VGA0, graphics, mouse, and keyboard. All subjects have 256MB of memory and 1 CPU. Virtual UART connection is configured between two of the PV Linux subjects.

```
host$ autoconfig mksrp target \
    --subject-pvlinux1=kernel=pvlinux-vmlinux-x86_64.bin,\
ramdisk=pvlinux-ramdisk-x86_64.img,VGA0,GRAPHICS*,MOUSE0,KEYBOARD,virtnet \
    --subject-pvlinux2=kernel=pvlinux-vmlinux-x86_64.bin,\
ramdisk=pvlinux-ramdisk-x86_64.img,virtnet=NAT,virtuart=pvlinux3,virtnet \
    --subject-pvlinux3=kernel=pvlinux-vmlinux-x86_64.bin,\
ramdisk=pvlinux-ramdisk-x86_64.img,virtuart=pvlinux2,virtnet \
    --output=/var/lib/tftpboot
```

- The first and third PV Linux Guest OSes will get a network IP address dynamically assigned to them by the DHCP server that the target connects to.
- The second PV Linux Guest OS network IP address will get the 10.1.0.2 IP address dynamically from VDS.
- Virtual serial connection is configured between the second and third PV Linux Guest OSes.

Para-Virtualized Linux and Fully Virtualized Subjects

This configuration describes a combination of FV and PV Guest OSes with virtual KVM, disk, network, USB and UART devices.

```
host$ autoconfig mksrp target \
    --subject-vds0=password=root123,loginconsoles=ttyS0,SERIAL* \
    --subject-fv1=virthd=Win7,ram=4g,cpus=2,virte1000=NET0,virte1000=fv2,virte1000=fv3,virtuart=fv2,virtohci \
    --subject-fv2=virthd=CentOS,ram=4g,cpus=2,virte1000=NET0,virte1000=fv1,virtuart=fv1,virtohci \
    --subject-fv3=virthd=Ubuntu,ram=4g,cpus=2,virte1000=NET0,virte1000=fv1,virtuart,virtohci \
    --subject-pvlinux=kernel=pvlinux-vmlinux-x86_64.bin,ramdisk=pvlinux-ramdisk-x86_64.img,\
virtnet=NET0,virtuart,loginconsoles=ttyVIRTUART0 \
    --output=/var/lib/tftpboot
```

- All the Guest OSes will get a network IP address dynamically assigned to them by the DHCP server that the target connects to.
- 1st and 2nd FV subjects are configured with a peer-to-peer network connection.
- 1st and 3rd FV subjects are configured with a peer-to-peer network connection.
- 1st and 2nd FV subjects are configured with a peer-to-peer virtual UART connection.
- 3rd FV subject has a virtual UART connection to VDS.
- PV Linux has a virtual UART connection to VDS. Login console will be created on this virtual UART; it will be possible to log in from VDS to PV Linux.

LynxSecure Applications (LSA)

This section provides examples of combining PV/FV LSAs with the FV Guest Operating Systems.

A FV Guest OS and LSA 32-bit PV configuration:

```
host$ autoconfig mksrp target \
    --subject-vds0=password=root123 \
    --subject-fv1=virthd=Win7,virtcd=CD-0000001F2-2.0.0.0-0,ram=4g,virtvga,virtkbdmouse \
    --subject-pvlsa0=SERIAL*,lsapath=$ENV_PREFIX/examples/lsa/lsa_pv32.bin \
    --output=/var/lib/tftpboot
```

A FV Guest OS and LSA 64-bit PV configuration:

```
host$ autoconfig mksrp target \
    --subject-vds0=password=root123 \
    --subject-fv1=virthd=Win7,virtcd=CD-0000001F2-2.0.0.0-0,ram=4g,virtvga,virtkbdmouse \
    --subject-pvlsa0=SERIAL*,lsapath=$ENV_PREFIX/examples/lsa/lsa_pv64.bin \
    --output=/var/lib/tftpboot
```

A FV Guest OS and LSA 32-bit FV configuration:

```
host$ autoconfig mksrp target \
    --subject-vds0=password=root123 \
    --subject-fv1=virthd=Win7,virtcd=CD-0000001F2-2.0.0.0-0,ram=4g,virtvga,virtkbdmouse \
    --subject-fvlsa0=SERIAL*,lsapath=$ENV_PREFIX/examples/lsa/lsa_fv.i386.bin \
    --output=/var/lib/tftpboot
```

A FV Guest OS and LSA 64-bit FV configuration:

```
host$ autoconfig mksrp target \
    --subject-vds0=password=root123 \
    --subject-fv1=virthd=Win7,virtcd=CD-0000001F2-2.0.0.0-0,ram=4g,virtvga,virtkbdmouse \
    --subject-fvlsa0=SERIAL*,lsapath=$ENV_PREFIX/examples/lsa/lsa_fv.bin \
    --output=/var/lib/tftpboot
```

UEFI based FV Guests

This section provides examples of UEFI based FV Guest Operating Systems.

FV UEFI Guest OS configuration:

```
host$ autoconfig mksrp target \
    --subject-vds0=password=root123 \
    --subject-fv1=virthd=Win10,virtcd=CD-0000001F2-2.0.0.0-0,ram=4g,virtvga,\
     virtkbdmouse,fwpath=OVMF.fd,virtlpc \
    --output=/var/lib/tftpboot
```

FV UEFI Guest OS and Legacy Guest OS configuration:

```
host$ autoconfig mksrp target \
    --subject-vds0=password=root123 \
    --subject-fv1=virthd=Win10,virtcd=CD-0000001F2-2.0.0.0-0,ram=4g,virtvga,virtkbdmouse, \
     fwpath=OVMF.fd,virtlpc \
    --subject-fv1=virthd=Win7,virtcd=CD-0000001F2-2.0.0.0-0,ram=4g,virtvga,virtkbdmouse \
    --output=/var/lib/tftpboot
```

LynxOS-178 2.2.4

This section provides description on how to boot LynxOS-178 images under LynxSecure.

Please refer to "LynxOS-178 v2.2.4 Release Notes" for more details on LynxOS-178 2.2.4.

LynxOS-178 2.2.4 as FV Guest OS on Zynq UltraScale+ MPSoC

```
autoconfig mksrp $TARGET \
--subject-fv_los178=devres=host,SERIAL0,NET3,USB0,PCIE0,ram=2g,\
kdipath=$KDIPATH \
-g
```

The command line above will create an SRP for Zynq board and will boot one LynxOS-178 guest with `SERIAL0`, USB, network controller #3 and PCI host bridge devices assigned to the subject.

Notes:

- `devres=host` instructs RIF to use the same MMIO ranges for GIC (Genaric Interrupt Controller) emulation as on the host, LynxOS-178 doesn't support FDT and have the device tree hardcoded, which means that GIC MMIO ranges in virtual environment must match the host's GIC MMIO ranges;
- LynxOS-178 supports up to 2G of address space, specifying `ram=` with values bigger than 2G will not have any effect.

APPENDIX B *Autoconfig Reference*

Commands Reference

Commands

`autoconfig collect`*`TARGET-NAME HOSTNAME|IP-ADDRESS|DTB-FILENAME [-g]`*

Collect the hardware information from the specified target system on the network.

`autoconfig delete`*`TARGET-NAME1 [TARGET-NAME2 ...]`*

Delete the entry for the specified target in the target database.

`autoconfig export`*`DIR [TARGET-NAME1 TARGET-NAME2 ...]`*

Export target database to the specified directory.

`autoconfig import`*`DIR [TARGET-NAME1 TARGET-NAME2 ...]`*

Import target database from the specified directory.

`autoconfig mkhcv`*`TARGET-NAME [--subject-name=...] [--option..]`*

Create LynxSecure Human-readable Configuration Vector (HCV).

`autoconfig mksrp`*`TARGET-NAME [--subject-name=...] [--option..] [DESTINATION]`*

Create bootable LynxSecure Runtime Package (SRP).

`autoconfig rename`*`TARGET-NAME-OLD TARGET-NAME-NEW`*

Rename a target entry in the database.

`autoconfig set`*`[--option=VALUE...] [-ggg]`*

Set the new default values for a list of options.

`autoconfig show`*`TARGET-NAME`*

Display the collected hardware information in the target database for a particular target.

`autoconfig targets`

Display names of all cached system configurations.

`autoconfig unset`*`[option...]`*

Reset the option values to their factory default values for a list of options.

Options Reference

Global Options

-arch=*x86_64|aarch64|ppc*

Select architecture for the SRP.

The option default value cannot be overwritten in the settings file.

-audit.fullbufferaction=*[MAINTMODE|SHUTDOWN|DROP|RESCHED[:PARAMETER]]*

Set the action to take if the LynxSecure audit buffer is full.

See also: section "System-wide Options When Creating an SRP or HCV" (page 23).

-audit.rule=*ACTION:EVENT-TYPE[:INITIATOR-NAME|any[:RECIPENT-NAME|any]]*

Setup action to take when the LynxSecure audit record is matched.

See also: section "System-wide Options When Creating an SRP or HCV" (page 23).

-audit.size=*NUMBER*

Set the maximum number of entries in the LynxSecure audit buffer; the value must be a power of two.

See also: section "System-wide Options When Creating an SRP or HCV" (page 23).

-build-dir=*DIR*

Custom build directory for XML/SRP (default is `$ENV_PREFIX/build/TARGET-NAME').

See also: Chapter 7, "*Subjects*" (page 45).

The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

-cbitfailaction=*ACTION*

Continuous Built-In Test (CBIT) action on failure.

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

-cbitinitparams=*OPTIONS-LIST*

Continuous Built-In Test (CBIT) initial paramaters.

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

-cbitpath=*FILENAME*

Continuous Built-In Test (CBIT) image path .

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

-cbits

Enable/disable `cbit` subjects creation.

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

-cbitwatchdog=*NUMBER*

Continuous Built-In Test (CBIT) watchdog timeout (ticks).

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`-convhcv=COMMAND`

Location/arguments of the `convhcv` tool.

`-develop.override-ram`

Override host RAM mappings.

The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

`-device-deps=[refs|no-refs][,...]`

Specify the treatment of device dependencies when assigning devices to subjects.

See also: section "System-wide Options When Creating an SRP or HCV" (page 23).

The option default value cannot be overwritten in the settings file.

`-device-list=[assignable|disabled][,...]` **(only on aarch64, ppc platforms)**

Specify the set of devices visible to the user.

See also: section "System-wide Options When Creating an SRP or HCV" (page 23).

The option default value cannot be overwritten in the settings file.

`-diagoutput=[vga|usb|serial[=SERIAL-DEVICE[:BAUD[:MAXBAUD]]]],...`

Enable/disable LynxSecure hypervisor diagnostic output.

See also: section "System-wide Options When Creating an SRP or HCV" (page 23).

`-dtbpath=FILENAME` **(only on aarch64, ppc platforms)**

The device tree blob path.

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`-explicit=[all|rmrr|memflows|systemram|bridgeinfo|gpa|gva|devresources|irq|ibit|vdev|vdevbdf|vf|skhheap|absclock|cpuid|cpuaffinity|misc|audit][,...]`

Enable/disable explicit specification of RMRRs, memory flows, system RAM, PCI bridges, guest physical addresses, guest virtual addresses, physical device resources, IRQs, IBIT subjects, detailed virtual devices, virtual device PCI bus/device/function address, SR-IOV virtual functions, SKH heap, absolute clock, hardware CPU IDs, CPU affinity, and miscellaneous items respectively. If this option is specified without any value, everything is explicit. `memflows' means that the Autoconfig Tool creates the optional memory flows that are otherwise created by RIF.

See also: Chapter 8, "*Physical Device Assignment to Subjects*" (page 57).

The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

`-extra-logs`

Collect logs and additional information from the target, such as dmesg, lspci, ACPI tables, /proc entries, etc.

See also: section "System-wide Options When Creating an SRP or HCV" (page 23).

The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

`-flowpolicy=[S2R|P2P|P2P+S2R]`

Assign resources to multiple partitions and produce the HCV with the specified configuration flow policy. If not specified, all resources are assigned to a single partition and the S2R policy is used. Using `--explicit' is recommended with this option; all resources created implicitly at runtime will be assigned to the system partition.

See also: section "<hcv> — Root Element" in *LynxSecure 6.3.0 Advanced Configuration Guide*.

`-force`

Ignore errors and proceed.

The option is experimental and may not work on all targets. The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

`-force-stolen-ram` (only on x86_64 platform)

Force using stolen RAM for VGA emulation.

The option is experimental and may not work on all targets. The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

`-forcepolling``=`*`DEVICE0[,DEVICE1,...]`*

List of devices which will use polling for interrupt emulation. Using this option will unnecessarily degrade devices' performance; usage of this option is not recommended except for testing purposes.

See also: Chapter 8, "*Physical Device Assignment to Subjects*" (page 57).

The option is experimental and may not work on all targets. The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

`-help`

Print help.

The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

`-ht` (only on x86_64 platform)

Enable/disable HyperThreading (if available).

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`-ibitfailaction``=`*`ACTION`*

Continuous Built-In Test (IBIT) action on failure.

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`-ibitinitparams``=`*`OPTIONS-LIST`*

Startup/Initiated Built-In Test (IBIT) initial paramaters.

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`-ibitpath``=`*`FILENAME`*

Startup/Initiated Built-In Test (IBIT) image path.

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`-ibitwatchdog``=`*`NUMBER`*

Startup/Initiated Built-In Test (IBIT) watchdog timeout (ticks).

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`-ibrs``=[disabled|subject|full]` **(only on x86_64 platform)**

Set the level of the Indirect Branch Restricted Speculation (IBRS) mitigations for the Spectre variant 2 attack. `disabled' means disable it in both hypervisor and subjects. `subject' (the default) means don't use it in the hypervisor, but make it available to subjects. And `full' means use it in the hypervisor and make it available to subjects. The hypervisor includes software mitigations for the Spectre variant 2 and doesn't normally need IBRS set to `full'; this option provides an increased security assurance. Note that IBRS comes with a significant performance penalty. It also requires hardware support; some processors may need a microcode update in order to support it.

See also: Chapter 4, "*Introduction to the Autoconfig Tool*" (page 15).

`-iommu`

Enable/disable switch for IOMMU.

See also: Chapter 8, "*Physical Device Assignment to Subjects*" (page 57).

`-jsonpath``=FILENAME`

The target hardware database path.

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`-keep-tmp-files`

Keep temporary files.

See also: section "System-wide Options When Creating an SRP or HCV" (page 23).

The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

`-l1tf``=[none|conditional|always]` **(only on x86_64 platform)**

Set the level of the mitigations for the L1TF attack. `none' means the mitigation is disabled. `conditional' (the default) means the mitigation is done conditionally, when the hypervisor believes it is required. And `always', which provides the highest security level, means that the mitigation is done every time when switching from hypervisor to subject code. The L1TF mitigation comes with a significant performance penalty; a microcode update may reduce that penalty. Note that L1TF attacks on the hypervisor by sibling Hyper-Threading threads cannot be mitigated. Users should consider not using Hyper-Threading on systems with CPUs vulnerable to L1TF attacks.

See also: Chapter 4, "*Introduction to the Autoconfig Tool*" (page 15).

`-mkcv``=COMMAND`

Location of the `mkcv` tool.

`-modules``=MODULES`

Modules to enable in the SRP, comma-separated.

See also: section "Enabling Additional Modules" in *LynxSecure 6.3.0 Advanced Configuration Guide*.

The option should be explicitly specified in the command line to take effect.

`-nopolling``=DEVICE0[,DEVICE1,...]`

List of devices for which interrupt emulation by the means of polling is disallowed.

See also: Chapter 8, "*Physical Device Assignment to Subjects*" (page 57).

The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

`-output``=FILENAME`

The SRP output binary path.

See also: Chapter 7, “*Subjects*” (page 45).

The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

`-rifargs``=OPT1[=VAL1] [OPT2=[VAL2]]...`

Add extra arguments for RIF.

See also: Chapter 5, “*Booting SRPs*” (page 27).

The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

`-runtimediag``=[none|fatalonly|all]`

Enable LynxSecure hypervisor Run-Time diagnostics (fatalonly means only for subject fatal faults).

See also: section “System-wide Options When Creating an SRP or HCV” (page 23).

`-sched.focus``=FOCUS-SUBJECT-VCPU-SHARE%`

Generate subject-in-focus schedule.

See also: Chapter 7, “*Subjects*” (page 45), Chapter 14, “*Virtual Keyboard Video Mouse (only x86 platform)*” (page 85).

The option should be explicitly specified in the command line to take effect.

`-sched.maintenance``SUBJECT-NAME1[,SUBJECT-NAME2,...]`

Add list of subjects to the maintenance schedule.

See also: Chapter 7, “*Subjects*” (page 45).

The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

`-sched.minorframe``=TICKS`

Defines the minimum minor frame length (ticks).

See also: Chapter 3, “*LynxSecure Features*” in *LynxSecure 6.3.0 Architecture Guide*.

`-sched.policy``=NAME:SUBJECT-NAME1[,SUBJECT-NAME2,...]`

Add custom operational scheduling policy.

See also: section “System-wide Options When Creating an SRP or HCV” (page 23).

The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

`-skhheap``=[ADDR,]SIZE`

The hypervisor heap size. The heap contains dynamic memory allocations. Heap memory is allocated by the hypervisor during startup only, but not after the hypervisor has begun executing subjects. The minimum required size of the heap depends on two things: the user-defined configuration and the target hardware specifics. At this time, the Autoconfig Tool doesn't have the ability to estimate the required heap size; instead, it uses a default value, which should be sufficient for most configurations. Should the default value be insufficient, please increase the heap size using this option. After a successful startup, LynxSecure hypervisor prints the actual final heap utilization. That information and this option may be used to trim the heap to the minimum required size. If the optional address is specified, the heap is created at the specified address in the host memory.

See also: section “System-wide Options When Creating an SRP or HCV” (page 23).

The option should be explicitly specified in the command line to take effect.

`-smidisable` **(only on x86_64 platform)**

Keep System Management Interrupts disabled/enabled.

See also: section “System-wide Options When Creating an SRP or HCV” (page 23).

The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

-srpdebug=*[all|skh|vds.debug|vds.usb|vds.net|vds.audio|vds.bde|vds.monitor|vds.kvm|fvs.debug|LSA.store][,....]*

Enable extra debugging in the SRP.

See also: section “System-wide Options When Creating an SRP or HCV” (page 23).

The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

-srpformat=*FORMAT*

Specifies the image format to pass to mkcv. See `mkcv -?' for more info.

See also: section “System-wide Options When Creating an SRP or HCV” (page 23).

-srploadaddr=*HPA*

The SRP load address.

See also: section “System-wide Options When Creating an SRP or HCV” (page 23).

The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

-subject-[cbit|fv|ibit|pvlinux|pvlsa|pvlynxos|vds]NAME=*[VIRTUAL-DEVICE0,VIRTUAL-DEVICE1,...],[PHYSICAL-DEVICE0,PHYSICAL-DEVICE1,...],[OPTION0,OPTION1,...]*

Specify a new subject.

See also: Chapter 7, “*Subjects*” (page 45).

The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

-tickspersec=*NUMBER*

Scheduling ticks per second.

See also: section “System-wide Options When Creating an SRP or HCV” (page 23).

The option should be explicitly specified in the command line to take effect.

-usbrule=*RULE* **(only on x86_64 platform)**

Add a new USB emulation rule to the end of the USB emulation rules list.

See also: Chapter 18, “*Virtual USB (only x86 platform)*” (page 99).

The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

-verbosity=*LOG_DEBUG|LOG_ERROR|LOG_INFO|LOG_VERBOSE|LOG_WARN|NUMBER*

Increase the verbosity of the tool.

See also: section “System-wide Options When Creating an SRP or HCV” (page 23).

The option default value cannot be overwritten in the settings file.

-version

Print version information.

The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

-virtusb-no-auto-rules (only on x86_64 platform)

Suppress adding automatically generated USB emulation rules.

See also: Chapter 18, “*Virtual USB (only x86 platform)*” (page 99).

-xmlname=*XMLNAME*

Custom name for XML/SRP (if different from target name).

See also: Chapter 7, "*Subjects*" (page 45).

The option default value cannot be overwritten in the settings file. The option should be explicitly specified in the command line to take effect.

Common Subject Options

DEVICE[NUMBER|*]?[#[NUMBER|*]][?][@]

The physical device name, as displayed in the target hardware list, to assign to the subject. If the 'DEVICE' syntax is used, the first unassigned device with the DEVICE prefix will be assigned to the subject. If the 'DEVICE*' syntax is used, all devices with the DEVICE prefix are assigned to the subject. If the 'DEVICE?' syntax is used, the DEVICE is optional, LynxSecure won't panic if it is missing. A 'DEVICE@' suffix may be used to skip resetting this device when its owner subject is restarted. If the device specification contains a number sign '#', the specification refers to an SR-IOV Virtual Function. The part of the specification before the colon identifies the Physical Function, and the number after the colon identifies the Virtual Function within that Physical Function. For example, NET0#2 refers to the SR-IOV virtual function #2 of the physical device NET0. If the VF number is omitted, the first available Virtual Function of the specified Physical Function is assigned. As a special case, an asterisk as the Virtual Function number indicates the assignment of the Physical Function along with all its Virtual Functions.

See also: Chapter 8, "*Physical Device Assignment to Subjects*" (page 57).

The option should be explicitly specified in the command line to take effect.

audit.irq=*IRQ-NUMBER*

Set audit IRQ, setting an IRQ also grants audit buffer read permissions to the subject. x86 valid ranges: [0..223], Arm valid ranges: [32..1019].

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

bootregion=*HPA:GPA:SIZE:TYPE:MODE[:NAME]*

Specify a custom boot memory region for a subject. This region should be pre-populated with a valid image before the subject is started. HPA and GPA can be either an address or `auto'. Addresses must be specified as a hexadecimal number starting with 0x. SIZE accepts decimal, hexadecimal, or decimal postfixed with K, M, or G. TYPE is one of LINUX_KERNEL, LINUX_RAMDISK, CPIO, or KDI. MODE can be one of RO, RW_RAM, or RW_RESERVED. In the case of RW_RAM, the guest image is not marked as reserved, and it is likely that the guest operating system will modify the contents of the memory region. Specify RW_RAM only if subject restart is not needed.

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

breakonstart (only on aarch64, ppc platforms)

Set an external debugger breakpoint at the subject start address (VCPU #0 only).

See also: Chapter 7, "*Subjects*" (page 45).

cpu_rng (only on x86_64, i386 platforms)

Allow/disallow access to the the CPU built-in hardware Random Number Generator.

See also: Chapter 4, "*Introduction to the Autoconfig Tool*" (page 15).

devicereset=*[true|false]*

Allow the subject to reset physical devices assigned to it using a generic mechanism, such as the FLR for PCI devices. If not specified, defaults to yes (true).

See also: Chapter 7, “*Subjects*” (page 45).

`devres``=`*`[auto|host|pool]`*

If set to `host', allocate device resources in the subject at their host locations, if possible. Use this setting if the subject uses resource addresses hardcoded for the host system. If set to `pool', allocate device resources from a common pool. Use this setting if the subject uses ACPI (x86) or the Devicetree (Arm) to find system resources. If unspecified or set to `auto', uses the platform default, which is `pool' for x86 and `host' for Arm.

See also: Chapter 7, “*Subjects*” (page 45).

`guestimage``=`*`NAME:FILENAME[:TYPE[:GPA[:GVA]]]`*

Specify a custom guest image for a subject. Supported types: PROGRAM (default), RESERVED, BOOT. GPA/ GVA is a guest physical/virtual address the guest image is mapped to, hexadecimal number with preceding '0x'.

See also: Chapter 7, “*Subjects*” (page 45).

The option should be explicitly specified in the command line to take effect.

`irq``=`*`IRQ:FROM-SUBJECT-NAME`*

Specify a custom IRQ number to inject into the subject. x86 valid ranges: [0..223], Arm valid ranges: [32..1019].

See also: Chapter 7, “*Subjects*” (page 45).

The option should be explicitly specified in the command line to take effect.

`isa``=`*`[32bit|64bit]`*

Explicitly specify the subject Instruction Set Architecture.

See also: Chapter 7, “*Subjects*” (page 45).

`memregion``=`*`NAME:SIZE[:TYPE[:MODE[:GPA[:GVA[:CACHING]]]]]`*

Specify a custom memory region for a subject. In case of a shared memory region, the region parameters will be overridden by the last occurrence of the memory region specification. The name of the region is global and may be used to refer to the same region from different subjects. Mode can be either RW or RO, with RW being the default. Supported types: PROGRAM, SHM, CMA, DMA, RESERVED, BITRESULTS, VDEVIO. PROGRAM creates available RAM. SHM creates memory shared between subjects; this is the default type. RESERVED creates a generic reserved memory region. CMA and DMA create Linux Contiguous Memory Allocator or regular DMA pools, respectively (only on Devicetree platforms). VDEVIO is a region for the hardware emulation. GPA/GVA is a guest physical/virtual address the memory region is mapped to, hexadecimal number with preceding '0x'. CACHING defines the region caching type: DEFAULT, UNCACHEABLE, WRITE-COMBINING, WRITE-THROUGH, WRITE-PROTECTED, WRITEBACK.

See also: Chapter 7, “*Subjects*” (page 45).

The option should be explicitly specified in the command line to take effect.

`pciecfg` (only on x86_64, i386 platforms)

Enable PCI Express in the subject. If this option is disabled, all assigned PCI devices will be visible as legacy PCI devices in the subject.

See also: Chapter 8, “*Physical Device Assignment to Subjects*” (page 57).

`realtime`

Real-time subject.

See also: Chapter 7, “*Subjects*” (page 45).

The option should be explicitly specified in the command line to take effect.

`role``=`*`[CBIT|IBIT|LSA|OTHER|VDS]`*

Explicitly specify the subject role.

See also: Chapter 7, “*Subjects*” (page 45).

`sanpath``=`*`FILENAME`*

Sanitization image path.

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`type``=`*`[cbit|fv|ibit|pvlinux|pvlsa|pvlynxos|vds]`*

Explicitly specify the subject type.

See also: Chapter 7, "*Subjects*" (page 45).

`vcpu.cache-capacity``=`*`NUMBER%%|SIZE|BITMASK|none`* **(only on x86_64, i386 platforms)**

Specify the cache partitioning settings for all virtual CPUs in the subject. The feature is disabled by default. Specifying the cache capacity in bytes or percents will result in an exclusive cache partition. If cache sharing between subjects is desired, use raw capacity bitmasks. All subjects without an explicit cache capacity specification will share one cache partition (the entire remaining cache capacity).

See also: Chapter 7, "*Subjects*" (page 45).

`wbinvd``=`*`[performance|native|secure_fast_dma|secure_fast_cache_flush]`* **(only on x86_64, i386 platforms)**

Enable Write Back and Invalidate Cache (WBINVD) CPU instruction handling.

See also: Chapter 7, "*Subjects*" (page 45).

FV Subject Options

`LSA.store``=`*`FILENAME`* **(only on x86_64, i386 platforms)**

Specify an LSA.store config, and enable LSA.store for this subject.

See also: Chapter 19, "*LSA.store Full Disk Encryption (only x86 platform)*" (page 113).

The option should be explicitly specified in the command line to take effect.

`apic` **(only on x86_64, i386 platforms)**

Specifies whether local APIC and I/O APIC emulation is provided for the subject.

See also: Chapter 7, "*Subjects*" (page 45).

`bdemode``=`*`AHCI|IDE|LEGACY`* **(only on x86_64, i386 platforms)**

Specify the virtual block device operation mode.

See also: Chapter 13, "*Virtual Block Device (only x86 platform)*" (page 81).

`bdetimeout``=`*`SECONDS`* **(only on x86_64, i386 platforms)**

Set up BDE timeout for the subject.

See also: Chapter 13, "*Virtual Block Device (only x86 platform)*" (page 81).

The option should be explicitly specified in the command line to take effect.

`bdetype``=`*`ICH7|ICH9|ISA|LYNX`* **(only on x86_64, i386 platforms)**

Specify the virtual block device controller.

See also: Chapter 13, "*Virtual Block Device (only x86 platform)*" (page 81).

`boottimeout``=`*`SECONDS`* **(only on x86_64, i386 platforms)**

Set up the boot timeout for the subject.

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`cpus`*`NUMBER[e]`*

Define the number of virtual CPUs for the subject. The 'e' postfix indicates "exclusive" VCPUs that will not share timeslices with any other VCPUs; 'max' instructs using the maximum number of physical CPUs available on the target system.

See also: Chapter 7, "*Subjects*" (page 45).

`flexcpus`*`DONOR[@ID][:DONOR[@ID]...]`*

Specify a set of flexible-scheduled virtual CPUs. These VCPUs are not made part of the schedule; for them to run, the donor VCPU must donate its CPU time. If not specified, the VCPU ID of the donor defaults to 0. The donor specification must refer to the owner of the scheduling minor frame (i.e. "original donor") in case there are multiple recipients of the donated time. The permissions to donate CPU time (and return the donation) must be specified explicitly.

See also: Chapter 7, "*Subjects*" (page 45).

`fvspath`*`=FILENAME`*

Subject FVS image path.

See also: Chapter 7, "*Subjects*" (page 45).

`fwpath`*`=FILENAME`* **(only on x86_64, i386 platforms)**

Subject firmware image path.

See also: Chapter 7, "*Subjects*" (page 45).

`identitymem`

Specifies that subject RAM is identity-mapped from the host address space to the guest address space.

See also: Chapter 8, "*Physical Device Assignment to Subjects*" (page 57).

The option should be explicitly specified in the command line to take effect.

`initialstate`*`=running|stopped`*

The initial state of the subject.

See also: Chapter 7, "*Subjects*" (page 45).

`initparams`*`=PARAMETERS-LIST`*

Override the default initparams for the subject.

See also: Chapter 7, "*Subjects*" (page 45).

`jump`*`=HEX`*

Specifies the jump address for the KDI.

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`kdidst`*`=HEX`*

Specifies the GPA to which the KDI will be copied before boot.

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`kdipath`*`=FILENAME`*

Specifies LynxOS kernel downloadable image (KDI) path.

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`kernel``=`*`FILENAME`*

Specifies the path to the GuestOS kernel.

See also: Chapter 7, "*Subjects*" (page 45).

`kernelconsole``=`*`TTY[:BAUD]`*

Add console settings to the Linux kernel command line (use colons instead of commas).

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`kernelearlyconsole``=`*`TTY[:BAUD]`*

Add early console settings to the Linux kernel command line (use colons instead of commas).

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`logger` (only on x86_64, i386 platforms)

Enable subject logging (logs are available in VDS).

`loginconsoles``=`*`TTY1:TTY2:...`*

Colon-separated list of output console device and options.

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`lsapath``=`*`FILENAME`*

LynxSecure Application image path.

See also: Chapter 7, "*Subjects*" (page 45).

`monitor` (only on x86_64, i386 platforms)

Enable the subject monitor (a facility that lets one modify FVS run-time settings from VDS).

`onfatalfault``=`*`ACTION`*

Triple fault action.

See also: Chapter 7, "*Subjects*" (page 45).

`orompath``=`*`FILENAME`* **(only on x86_64, i386 platforms)**

OptionROM image path.

See also: Chapter 8, "*Physical Device Assignment to Subjects*" (page 57).

The option should be explicitly specified in the command line to take effect.

`perm``=`*`[readsched|writesched|halt|restart|guesthypercall|maintenance|skdb|absclock|readcal|writecal|flexreturn|readaudit|writeaudit|bit]:...`*

Set global permissions, colon-separated.

See also: Chapter 7, "*Subjects*" (page 45).

`pip`*`[=SECURE-LABEL:SECURE-LABEL-COLOR:FRAMEBUFFER-SIZE[k|K|m|M]]`* **(only on x86_64, i386 platforms)**

Used for passing secure label and its color as well as framebuffer size options.

See also: Chapter 14, "*Virtual Keyboard Video Mouse (only x86 platform)*" (page 85).

The option should be explicitly specified in the command line to take effect.

`pipmanager` (only on x86_64, i386 platforms)

Define this subject as the manager for the Picture in Picture feature.

See also: Chapter 14, "*Virtual Keyboard Video Mouse (only x86 platform)*" (page 85).

The option should be explicitly specified in the command line to take effect.

`ram``=`*`size[K|M|G]`*

Explicitly specify the total usable subject RAM size.

See also: Chapter 7, "*Subjects*" (page 45).

`ramdisk``=`*`FILENAME`*

Specifies the path to the GuestOS RAM disk.

See also: Chapter 7, "*Subjects*" (page 45).

`rdt.monitor` **(only on x86_64 platform)**

Grants access to Intel's RDT monitoring features.

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`ropagesize``=`*`size[K|M|G]`*

Explicitly specify the subject ROPAGE size.

See also: Chapter 7, "*Subjects*" (page 45).

`subjperm``=`*`SUBJECT:[start|stop|restart|suspend|resume|status|flexdonate|all-subject-states]:...`*

Set permissions to act on another subject.

See also: Chapter 7, "*Subjects*" (page 45).

`virtac97` **(only on x86_64, i386 platforms)**

Add a virtual Intel Corporation 82801AA AC'97 Audio Controller.

See also: Chapter 12, "*Virtual Audio (only x86 platform)*" (page 79).

The option should be explicitly specified in the command line to take effect.

`virtcd``=`*`[CDROM|FTP-URL|NFS-URL]`* **(only on x86_64, i386 platforms)**

Specify the virtual CD-ROM.

See also: Chapter 13, "*Virtual Block Device (only x86 platform)*" (page 81).

The option should be explicitly specified in the command line to take effect.

`virte1000`*`[=SUBJECT-NAME|NAT[:A.B]|NETx]|[@macaddr=XX:XX:XX:XX:XX:XX@peer=SUBJECT-NAME@mode=NAT@source=[NET0:...]@bdf=00:HEX.DEC@irq=IRQ]`* **(only on x86_64, i386 platforms)**

Add a virtual Intel Corporation 82540EM Gigabit Ethernet Controller (E1000).

See also: Chapter 15, "*Virtual Network (only x86 platform)*" (page 89).

The option should be explicitly specified in the command line to take effect.

`virtehci`*`[:ndp=NUMBER:[alias=ALIAS]]`* **(only on x86_64, i386 platforms)**

Add a virtual EHCI Controller.

See also: Chapter 18, "*Virtual USB (only x86 platform)*" (page 99).

The option should be explicitly specified in the command line to take effect.

`virtfifo`*`[=SUBJECT-NAME[:FIFO-NAME]]`*

Add an inter-subject fifo.

See also: Chapter 17, "*Virtual Shared Memory*" (page 97).

The option should be explicitly specified in the command line to take effect.

`virtgic` **(only on aarch64 platform)**

Add a virtual generic interrupt controller v2.

See also: .

The option should be explicitly specified in the command line to take effect.

`virthd`*`=[PARTITION-NAME|GPT-LABEL|NFS-URL|FILENAME]`* **(only on x86_64, i386 platforms)**

Specify the virtual HDD.

See also: Chapter 13, "*Virtual Block Device (only x86 platform)*" (page 81).

The option should be explicitly specified in the command line to take effect.

`virthda` **(only on x86_64, i386 platforms)**

Add a virtual Intel Corporation 82801JD/DO (ICH10 Family) HD Audio Controller.

See also: Chapter 12, "*Virtual Audio (only x86 platform)*" (page 79).

The option should be explicitly specified in the command line to take effect.

`virtkbdmouse` **(only on x86_64, i386 platforms)**

Add a virtual keyboard and mouse.

See also: Chapter 14, "*Virtual Keyboard Video Mouse (only x86 platform)*" (page 85).

The option should be explicitly specified in the command line to take effect.

`virtlpc` **(only on x86_64, i386 platforms)**

Add a virtual PCI-to-LPC bridge.

See also: Chapter 10, "*Virtual PCI-to-LPC bridge (only x86 platform)*" (page 75).

The option should be explicitly specified in the command line to take effect.

`virtne2000`*`[=SUBJECT-NAME|NAT[:A.B]|NETx]|[@macaddr=XX:XX:XX:XX:XX:XX@peer=SUBJECT-NAME@mode=NAT@source=[NET0:...]@bdf=00:HEX.DEC@irq=IRQ]`* **(only on x86_64, i386 platforms)**

Add a virtual Realtek Semiconductor Co., Ltd. RTL-8029(AS) Ethernet Controller (NE2000).

See also: Chapter 15, "*Virtual Network (only x86 platform)*" (page 89).

The option should be explicitly specified in the command line to take effect.

`virtnet`*`[=SUBJECT-NAME|NAT[:A.B]|NETx]|[@type=[virte1000|virtne2000]@ipaddr=A.B.C.D./SUBNET@macaddr=XX:XX:XX:XX:XX:XX@peer=SUBJECT-NAME@mode=NAT@source=[NET0:...]@bdf=00:HEX.DEC@irq=IRQ]`* **(only on x86_64, i386 platforms)**

Add a virtual network interface.

See also: Chapter 15, "*Virtual Network (only x86 platform)*" (page 89).

The option should be explicitly specified in the command line to take effect.

`virtohci`*`[:ndp=NUMBER:[alias=ALIAS]]`* **(only on x86_64, i386 platforms)**

Add a virtual OHCI Controller.

See also: Chapter 18, "*Virtual USB (only x86 platform)*" (page 99).

The option should be explicitly specified in the command line to take effect.

`virtshmem`*`[=SUBJECT-NAME[:CONFIG-STRING:CONFIG-STRING-PEER[:SHMEM-NAME]]]`*

Add a virtual shared memory controller.

See also: Chapter 17, "*Virtual Shared Memory*" (page 97).

The option should be explicitly specified in the command line to take effect.

`virttimer` **(only on aarch64 platform)**

Add a virtual timer controller.

See also: .

The option should be explicitly specified in the command line to take effect.

`virtuart``[=SUBJECT-NAME|:peer=SUBJECT-NAME[:type=[16550A|16750][:ioport=HEX[:irq=PIC-IRQ]]|:config={com1|com2|com3|com4}]]` **(only on x86_64, i386 platforms)**

Add a virtual UART controller.

See also: Chapter 16, "*Virtual UART (only x86 platform)*" (page 95).

The option should be explicitly specified in the command line to take effect.

`virtuhci``[:ndp=NUMBER:[alias=ALIAS]]` **(only on x86_64, i386 platforms)**

Add a virtual UHCI Controller.

See also: Chapter 18, "*Virtual USB (only x86 platform)*" (page 99).

The option should be explicitly specified in the command line to take effect.

`virtvga` **(only on x86_64, i386 platforms)**

Add a virtual VGA.

See also: Chapter 14, "*Virtual Keyboard Video Mouse (only x86 platform)*" (page 85).

The option should be explicitly specified in the command line to take effect.

PV Linux Subject Options

`bdemode``=AHCI|IDE|LEGACY` **(only on x86_64, i386 platforms)**

Specify the virtual block device operation mode.

See also: Chapter 13, "*Virtual Block Device (only x86 platform)*" (page 81).

`bdetype``=ICH7|ICH9|ISA|LYNX` **(only on x86_64, i386 platforms)**

Specify the virtual block device controller.

See also: Chapter 13, "*Virtual Block Device (only x86 platform)*" (page 81).

`cpus``NUMBER[e]`

Define the number of virtual CPUs for the subject. The 'e' postfix indicates "exclusive" VCPUs that will not share timeslices with any other VCPUs; 'max' instructs using the maximum number of physical CPUs available on the target system.

See also: Chapter 7, "*Subjects*" (page 45).

`dnsdomain``=NAME1[:NAME2...]` **(only on x86_64, i386 platforms)**

The DNS domain name list.

See also: Chapter 15, "*Virtual Network (only x86 platform)*" (page 89).

The option should be explicitly specified in the command line to take effect.

`dnsnameserver``=NAME1[:NAME2...]` **(only on x86_64, i386 platforms)**

The list of DNS nameservers.

See also: Chapter 15, "*Virtual Network (only x86 platform)*" (page 89).

The option should be explicitly specified in the command line to take effect.

`dnssearch``=NAME1[:NAME2...]` **(only on x86_64, i386 platforms)**

The DNS domain search list.

See also: Chapter 15, "*Virtual Network (only x86 platform)*" (page 89).

The option should be explicitly specified in the command line to take effect.

`flexcpus`*`DONOR[@ID][:DONOR[@ID]...]`*

Specify a set of flexible-scheduled virtual CPUs. These VCPUs are not made part of the schedule; for them to run, the donor VCPU must donate its CPU time. If not specified, the VCPU ID of the donor defaults to 0. The donor specification must refer to the owner of the scheduling minor frame (i.e. "original donor") in case there are multiple recipients of the donated time. The permissions to donate CPU time (and return the donation) must be specified explicitly.

See also: Chapter 7, "*Subjects*" (page 45).

`gateway`*`=NAME`* **(only on x86_64, i386 platforms)**

The default gateway.

See also: Chapter 15, "*Virtual Network (only x86 platform)*" (page 89).

The option should be explicitly specified in the command line to take effect.

`initialstate`*`=running|stopped`*

The initial state of the subject.

See also: Chapter 7, "*Subjects*" (page 45).

`initparams`*`=PARAMETERS-LIST`*

Override the default initparams for the subject.

See also: Chapter 7, "*Subjects*" (page 45).

`kernel`*`=FILENAME`*

Specifies the path to the GuestOS kernel.

See also: Chapter 7, "*Subjects*" (page 45).

`kernelconsole`*`=TTY[:BAUD]`*

Add console settings to the Linux kernel command line (use colons instead of commas).

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`kernelearlyconsole`*`=TTY[:BAUD]`*

Add early console settings to the Linux kernel command line (use colons instead of commas).

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`logger` **(only on x86_64, i386 platforms)**

Enable subject logging (logs are available in VDS).

`loginconsoles`*`=TTY1:TTY2:...`*

Colon-separated list of output console device and options.

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`onfatalfault`*`=ACTION`*

Triple fault action.

See also: Chapter 7, "*Subjects*" (page 45).

`password` **(only on x86_64, i386 platforms)**

The default password for the root account in vds0 subject.

See also: Chapter 9, “*Virtual Device Server (only x86 platform)*” (page 65).

The option should be explicitly specified in the command line to take effect.

`perm``=`*`[readsched|writesched|halt|restart|guesthypercall|maintenance|skdb|absclock|readcal|writecal|flexreturn|readaudit|writeaudit|bit]:...`*

Set global permissions, colon-separated.

See also: Chapter 7, “*Subjects*” (page 45).

`persistentstorage``=`*`GPT-LABEL`* **(only on x86_64, i386 platforms)**

GPT label of the persistent storage in the VDS subject.

See also: Chapter 9, “*Virtual Device Server (only x86 platform)*” (page 65).

The option should be explicitly specified in the command line to take effect.

`ram``=`*`size[K|M|G]`*

Explicitly specify the total usable subject RAM size.

See also: Chapter 7, “*Subjects*” (page 45).

`ramdisk``=`*`FILENAME`*

Specifies the path to the GuestOS RAM disk.

See also: Chapter 7, “*Subjects*” (page 45).

`rdt.monitor` **(only on x86_64 platform)**

Grants access to Intel's RDT monitoring features.

See also: Chapter 7, “*Subjects*” (page 45).

The option should be explicitly specified in the command line to take effect.

`ropagesize``=`*`size[K|M|G]`*

Explicitly specify the subject ROPAGE size.

See also: Chapter 7, “*Subjects*” (page 45).

`squashoverlayfs``=`*`GPT-LABEL[:FILE-PATH]`* **(only on x86_64, i386 platforms)**

Location of the VDS subject overlay filesystem.

See also: Chapter 9, “*Virtual Device Server (only x86 platform)*” (page 65).

The option should be explicitly specified in the command line to take effect.

`squashrootfs``=`*`GPT-LABEL[:FILE-PATH]`* **(only on x86_64, i386 platforms)**

Location of the VDS subject root filesystem.

See also: Chapter 9, “*Virtual Device Server (only x86 platform)*” (page 65).

The option should be explicitly specified in the command line to take effect.

`subjperm``=`*`SUBJECT:[start|stop|restart|suspend|resume|status|flexdonate|all-subject-states]:...`*

Set permissions to act on another subject.

See also: Chapter 7, “*Subjects*” (page 45).

`virtcd``=`*`[CDROM|FTP-URL|NFS-URL]`* **(only on x86_64, i386 platforms)**

Specify the virtual CD-ROM.

See also: Chapter 13, “*Virtual Block Device (only x86 platform)*” (page 81).

The option should be explicitly specified in the command line to take effect.

`virtfifo``[=SUBJECT-NAME[:FIFO-NAME]]`

Add an inter-subject fifo.

See also: Chapter 17, "*Virtual Shared Memory*" (page 97).

The option should be explicitly specified in the command line to take effect.

`virtgic` **(only on aarch64 platform)**

Add a virtual generic interrupt controller v2.

See also: .

The option should be explicitly specified in the command line to take effect.

`virthd``=[PARTITION-NAME|GPT-LABEL|NFS-URL|FILENAME]` **(only on x86_64, i386 platforms)**

Specify the virtual HDD.

See also: Chapter 13, "*Virtual Block Device (only x86 platform)*" (page 81).

The option should be explicitly specified in the command line to take effect.

`virtnet``[=SUBJECT-NAME|NAT[:A.B]|NETx]|[@type=[virte1000|virtne2000]@ipaddr=A.B.C.D./SUBNET@macaddr=XX:XX:XX:XX:XX:XX@peer=SUBJECT-NAME@mode=NAT@source=[NET0:...]@bdf=00:HEX.DEC@irq=IRQ]` **(only on x86_64, i386 platforms)**

Add a virtual network interface.

See also: Chapter 15, "*Virtual Network (only x86 platform)*" (page 89).

The option should be explicitly specified in the command line to take effect.

`virtshmem``[=SUBJECT-NAME[:CONFIG-STRING:CONFIG-STRING-PEER[:SHMEM-NAME]]]`

Add a virtual shared memory controller.

See also: Chapter 17, "*Virtual Shared Memory*" (page 97).

The option should be explicitly specified in the command line to take effect.

`virttimer` **(only on aarch64 platform)**

Add a virtual timer controller.

See also: .

The option should be explicitly specified in the command line to take effect.

`virtuart``[=SUBJECT-NAME|:peer=SUBJECT-NAME[:type=[16550A|16750][:ioport=HEX[:irq=PIC-IRQ]]|:config={com1|com2|com3|com4}]]` **(only on x86_64, i386 platforms)**

Add a virtual UART controller.

See also: Chapter 16, "*Virtual UART (only x86 platform)*" (page 95).

The option should be explicitly specified in the command line to take effect.

PV LSA Subject Options

`cpus``NUMBER[e]`

Define the number of virtual CPUs for the subject. The 'e' postfix indicates "exclusive" VCPUs that will not share timeslices with any other VCPUs; 'max' instructs using the maximum number of physical CPUs available on the target system.

See also: Chapter 7, "*Subjects*" (page 45).

`flexcpus``DONOR[@ID][:DONOR[@ID]...]`

Specify a set of flexible-scheduled virtual CPUs. These VCPUs are not made part of the schedule; for them to run, the donor VCPU must donate its CPU time. If not specified, the VCPU ID of the donor defaults to 0. The donor specification must refer to the owner of the scheduling minor frame (i.e. "original donor") in case there are multiple recipients of the donated time. The permissions to donate CPU time (and return the donation) must be specified explicitly.

See also: Chapter 7, "*Subjects*" (page 45).

`initialstate``=`*`running|stopped`*

The initial state of the subject.

See also: Chapter 7, "*Subjects*" (page 45).

`initparams``=`*`PARAMETERS-LIST`*

Override the default initparams for the subject.

See also: Chapter 7, "*Subjects*" (page 45).

`logger` **(only on x86_64, i386 platforms)**

Enable subject logging (logs are available in VDS).

`lsapath``=`*`FILENAME`*

LynxSecure Application image path.

See also: Chapter 7, "*Subjects*" (page 45).

`onfatalfault``=`*`ACTION`*

Triple fault action.

See also: Chapter 7, "*Subjects*" (page 45).

`perm``=`*`[readsched|writesched|halt|restart|guesthypercall|maintenance|skdb|absclock|readcal|writecal|flexreturn|readaudit|writeaudit|bit]:...`*

Set global permissions, colon-separated.

See also: Chapter 7, "*Subjects*" (page 45).

`ram``=`*`size[K|M|G]`*

Explicitly specify the total usable subject RAM size.

See also: Chapter 7, "*Subjects*" (page 45).

`rdt.monitor` **(only on x86_64 platform)**

Grants access to Intel's RDT monitoring features.

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`ropagesize``=`*`size[K|M|G]`*

Explicitly specify the subject ROPAGE size.

See also: Chapter 7, "*Subjects*" (page 45).

`subjperm``=`*`SUBJECT:[start|stop|restart|suspend|resume|status|flexdonate|all-subject-states]:...`*

Set permissions to act on another subject.

See also: Chapter 7, "*Subjects*" (page 45).

`virtgic` **(only on aarch64 platform)**

Add a virtual generic interrupt controller v2.

See also: .

The option should be explicitly specified in the command line to take effect.

`virttimer` **(only on aarch64 platform)**

Add a virtual timer controller.

See also: .

The option should be explicitly specified in the command line to take effect.

PV LynxOS Subject Options

`bdemode``=`*`AHCI|IDE|LEGACY`* **(only on x86_64, i386 platforms)**

Specify the virtual block device operation mode.

See also: Chapter 13, "*Virtual Block Device (only x86 platform)*" (page 81).

`bdetype``=`*`ICH7|ICH9|ISA|LYNX`* **(only on x86_64, i386 platforms)**

Specify the virtual block device controller.

See also: Chapter 13, "*Virtual Block Device (only x86 platform)*" (page 81).

`cpus`*`NUMBER[e]`*

Define the number of virtual CPUs for the subject. The 'e' postfix indicates "exclusive" VCPUs that will not share timeslices with any other VCPUs; 'max' instructs using the maximum number of physical CPUs available on the target system.

See also: Chapter 7, "*Subjects*" (page 45).

`flexcpus`*`DONOR[@ID][:DONOR[@ID]...]`*

Specify a set of flexible-scheduled virtual CPUs. These VCPUs are not made part of the schedule; for them to run, the donor VCPU must donate its CPU time. If not specified, the VCPU ID of the donor defaults to 0. The donor specification must refer to the owner of the scheduling minor frame (i.e. "original donor") in case there are multiple recipients of the donated time. The permissions to donate CPU time (and return the donation) must be specified explicitly.

See also: Chapter 7, "*Subjects*" (page 45).

`initialstate``=`*`running|stopped`*

The initial state of the subject.

See also: Chapter 7, "*Subjects*" (page 45).

`initparams``=`*`PARAMETERS-LIST`*

Override the default initparams for the subject.

See also: Chapter 7, "*Subjects*" (page 45).

`kdipath``=`*`FILENAME`*

Specifies LynxOS kernel downloadable image (KDI) path.

See also: Chapter 7, "*Subjects*" (page 45).

The option should be explicitly specified in the command line to take effect.

`onfatalfault``=`*`ACTION`*

Triple fault action.

See also: Chapter 7, "*Subjects*" (page 45).

`perm``=`*`[readsched|writesched|halt|restart|guesthypercall|maintenance|skdb|absclock|readcal|writecal|flexreturn|readaudit|writeaudit|bit]:...`*

Set global permissions, colon-separated.

See also: Chapter 7, "*Subjects*" (page 45).

ram=*size[K|M|G]*

Explicitly specify the total usable subject RAM size.

See also: Chapter 7, “*Subjects*” (page 45).

rdt.monitor (only on x86_64 platform)

Grants access to Intel's RDT monitoring features.

See also: Chapter 7, “*Subjects*” (page 45).

The option should be explicitly specified in the command line to take effect.

ropagesize=*size[K|M|G]*

Explicitly specify the subject ROPAGE size.

See also: Chapter 7, “*Subjects*” (page 45).

subjperm=*SUBJECT:[start|stop|restart|suspend|resume|status|flexdonate|all-subject-states]:...*

Set permissions to act on another subject.

See also: Chapter 7, “*Subjects*” (page 45).

virtcd=*[CDROM|FTP-URL|NFS-URL]* **(only on x86_64, i386 platforms)**

Specify the virtual CD-ROM.

See also: Chapter 13, “*Virtual Block Device (only x86 platform)*” (page 81).

The option should be explicitly specified in the command line to take effect.

virtfifo*[=SUBJECT-NAME[:FIFO-NAME]]*

Add an inter-subject fifo.

See also: Chapter 17, “*Virtual Shared Memory*” (page 97).

The option should be explicitly specified in the command line to take effect.

virtgic (only on aarch64 platform)

Add a virtual generic interrupt controller v2.

See also: .

The option should be explicitly specified in the command line to take effect.

virthd=*[PARTITION-NAME|GPT-LABEL|NFS-URL|FILENAME]* **(only on x86_64, i386 platforms)**

Specify the virtual HDD.

See also: Chapter 13, “*Virtual Block Device (only x86 platform)*” (page 81).

The option should be explicitly specified in the command line to take effect.

virtnet*[=SUBJECT-NAME|NAT[:A.B]|NETx]|[@type=[virte1000|virtne2000]@ipaddr=A.B.C.D./SUBNET@macaddr=XX:XX:XX:XX:XX:XX@peer=SUBJECT-NAME@mode=NAT@source=[NET0:...]@bdf=00:HEX.DEC@irq=IRQ]* **(only on x86_64, i386 platforms)**

Add a virtual network interface.

See also: Chapter 15, “*Virtual Network (only x86 platform)*” (page 89).

The option should be explicitly specified in the command line to take effect.

virttimer (only on aarch64 platform)

Add a virtual timer controller.

See also: .

The option should be explicitly specified in the command line to take effect.

`virtuart`*`[=SUBJECT-NAME|:peer=SUBJECT-NAME[:type=[16550A|16750][:ioport=HEX[:irq=PIC-IRQ]]|:config={com1|com2|com3|com4}]]`* **(only on x86_64, i386 platforms)**

Add a virtual UART controller.

See also: Chapter 16, "*Virtual UART (only x86 platform)*" (page 95).

The option should be explicitly specified in the command line to take effect.

VDS Subject Options

`cpus`*`NUMBER[e]`*

Define the number of virtual CPUs for the subject. The 'e' postfix indicates "exclusive" VCPUs that will not share timeslices with any other VCPUs; 'max' instructs using the maximum number of physical CPUs available on the target system.

See also: Chapter 7, "*Subjects*" (page 45).

`dnsdomain`*`=NAME1[:NAME2...]`* **(only on x86_64, i386 platforms)**

The DNS domain name list.

See also: Chapter 15, "*Virtual Network (only x86 platform)*" (page 89).

The option should be explicitly specified in the command line to take effect.

`dnsnameserver`*`=NAME1[:NAME2...]`* **(only on x86_64, i386 platforms)**

The list of DNS nameservers.

See also: Chapter 15, "*Virtual Network (only x86 platform)*" (page 89).

The option should be explicitly specified in the command line to take effect.

`dnssearch`*`=NAME1[:NAME2...]`* **(only on x86_64, i386 platforms)**

The DNS domain search list.

See also: Chapter 15, "*Virtual Network (only x86 platform)*" (page 89).

The option should be explicitly specified in the command line to take effect.

`flexcpus`*`DONOR[@ID][:DONOR[@ID]...]`*

Specify a set of flexible-scheduled virtual CPUs. These VCPUs are not made part of the schedule; for them to run, the donor VCPU must donate its CPU time. If not specified, the VCPU ID of the donor defaults to 0. The donor specification must refer to the owner of the scheduling minor frame (i.e. "original donor") in case there are multiple recipients of the donated time. The permissions to donate CPU time (and return the donation) must be specified explicitly.

See also: Chapter 7, "*Subjects*" (page 45).

`gateway`*`=NAME`* **(only on x86_64, i386 platforms)**

The default gateway.

See also: Chapter 15, "*Virtual Network (only x86 platform)*" (page 89).

The option should be explicitly specified in the command line to take effect.

`initparams`*`=PARAMETERS-LIST`*

Override the default initparams for the subject.

See also: Chapter 7, "*Subjects*" (page 45).

`keeplogs`*`[=SIZE]`* **(only on x86_64, i386 platforms)**

Sets the maximum amount of RAM used by logging. Can be specified as an absolute size in bytes, or as a percentage of total RAM. If specified, log rotation will also be disabled.

See also: Chapter 9, “*Virtual Device Server (only x86 platform)*” (page 65).

The option should be explicitly specified in the command line to take effect.

`kernel``=`*`FILENAME`*

Specifies the path to the GuestOS kernel.

See also: Chapter 7, “*Subjects*” (page 45).

`kernelconsole``=`*`TTY[:BAUD]`*

Add console settings to the Linux kernel command line (use colons instead of commas).

See also: Chapter 7, “*Subjects*” (page 45).

The option should be explicitly specified in the command line to take effect.

`kernelearlyconsole``=`*`TTY[:BAUD]`*

Add early console settings to the Linux kernel command line (use colons instead of commas).

See also: Chapter 7, “*Subjects*” (page 45).

The option should be explicitly specified in the command line to take effect.

`loginconsoles``=`*`TTY1:TTY2:...`*

Colon-separated list of output console device and options.

See also: Chapter 7, “*Subjects*” (page 45).

The option should be explicitly specified in the command line to take effect.

`onfatalfault``=`*`ACTION`*

Triple fault action.

See also: Chapter 7, “*Subjects*” (page 45).

`password` **(only on x86_64, i386 platforms)**

The default password for the root account in vds0 subject.

See also: Chapter 9, “*Virtual Device Server (only x86 platform)*” (page 65).

The option should be explicitly specified in the command line to take effect.

`perm``=`*`[readsched|writesched|halt|restart|guesthypercall|maintenance|skdb|absclock|readcal|writecal|flexreturn|readaudit|writeaudit|bit]:...`*

Set global permissions, colon-separated.

See also: Chapter 7, “*Subjects*” (page 45).

`persistentstorage``=`*`GPT-LABEL`* **(only on x86_64, i386 platforms)**

GPT label of the persistent storage in the VDS subject.

See also: Chapter 9, “*Virtual Device Server (only x86 platform)*” (page 65).

The option should be explicitly specified in the command line to take effect.

`ram``=`*`size[K|M|G]`*

Explicitly specify the total usable subject RAM size.

See also: Chapter 7, “*Subjects*” (page 45).

`ramdisk``=`*`FILENAME`*

Specifies the path to the GuestOS RAM disk.

See also: Chapter 7, “*Subjects*” (page 45).

`rdt.monitor` **(only on x86_64 platform)**

Grants access to Intel's RDT monitoring features.

See also: Chapter 7, “*Subjects*” (page 45).

The option should be explicitly specified in the command line to take effect.

`ropagesize``=`*`size[K|M|G]`*

Explicitly specify the subject ROPAGE size.

See also: Chapter 7, “*Subjects*” (page 45).

`squashoverlayfs``=`*`GPT-LABEL[:FILE-PATH]`* **(only on x86_64, i386 platforms)**

Location of the VDS subject overlay filesystem.

See also: Chapter 9, “*Virtual Device Server (only x86 platform)*” (page 65).

The option should be explicitly specified in the command line to take effect.

`squashrootfs``=`*`GPT-LABEL[:FILE-PATH]`* **(only on x86_64, i386 platforms)**

Location of the VDS subject root filesystem.

See also: Chapter 9, “*Virtual Device Server (only x86 platform)*” (page 65).

The option should be explicitly specified in the command line to take effect.

`subjperm``=`*`SUBJECT:[start|stop|restart|suspend|resume|status|flexdonate|all-subject-states]:...`*

Set permissions to act on another subject.

See also: Chapter 7, “*Subjects*” (page 45).

`tpmsecret``=`*`FILENAME`* **(only on x86_64, i386 platforms)**

Specifies the path to the sealed secret key for the target.

See also: Chapter 7, “*Subjects*” (page 45).

The option should be explicitly specified in the command line to take effect.

`virtfifo`*`[=SUBJECT-NAME[:FIFO-NAME]]`*

Add an inter-subject fifo.

See also: Chapter 17, “*Virtual Shared Memory*” (page 97).

The option should be explicitly specified in the command line to take effect.

`virtgic` **(only on aarch64 platform)**

Add a virtual generic interrupt controller v2.

See also: .

The option should be explicitly specified in the command line to take effect.

`virtnet`*`[=SUBJECT-NAME|NAT[:A.B]|NETx]|[@type=[virte1000|virtne2000]@ipaddr=A.B.C.D./SUBNET@macaddr=XX:XX:XX:XX:XX:XX@peer=SUBJECT-NAME@mode=NAT@source=[NET0:...]@bdf=00:HEX.DEC@irq=IRQ]`* **(only on x86_64, i386 platforms)**

Add a virtual network interface.

See also: Chapter 15, “*Virtual Network (only x86 platform)*” (page 89).

The option should be explicitly specified in the command line to take effect.

`virtshmem`*`[=SUBJECT-NAME[:CONFIG-STRING:CONFIG-STRING-PEER[:SHMEM-NAME]]]`*

Add a virtual shared memory controller.

See also: Chapter 17, “*Virtual Shared Memory*” (page 97).

The option should be explicitly specified in the command line to take effect.

`virttimer` (only on aarch64 platform)

Add a virtual timer controller.

See also: .

The option should be explicitly specified in the command line to take effect.

APPENDIX C *Introduction to LynxSecure License Management*

This section is relevant to LynxSecure® releases 6.0 and newer that use the Lynx License Management Software based on FLEXnet Publisher version 11.14.1.3 and higher.

The goal of the Lynx Software Technologies, Inc. License Management Software is to enforce the Product License Agreement by controlling and monitoring access to licensed products. Lynx Software Technologies, Inc. has chosen to utilize the FLEXnet Publisher licensing software (previously known as FLEXlm) as the basis for the Lynx Software Technologies, Inc. License Management software. The License Management process requires licensed products to communicate with the license server and check out a license before the product can be used.

Lynx Software Technologies, Inc. products that use the FLEXnet Publisher licensing software are provided license keys on an annual basis. License keys generally expire every 12 months and require renewal. The license key renewal will most often be triggered by the product's annual support renewal date.

The FLEXnet Publisher license server keeps track of how many licenses have been purchased and checks them out to the users wanting to start a protected product. For example, each developer who wants to run Lynx Software Technologies, Inc. CDK tools shall start a development session by running the Lynx Software Technologies, Inc. Session Manager (`lwsmgr`). The Session Manager sends the request to the FLEXnet Publisher license server. If a license is available, the FLEXnet Publisher license server will check out a license for that developer. Once a developer has successfully started a development session via `lwsmgr`, CDK tools can be run. At the conclusion of a development session, the user checks the license back in using the Session Manager tool. The license is now available to other users.

Please note that some Lynx Software Technologies, Inc. products and applications will check out a license directly from the FLEXnet Publisher license server.

The remaining sections of this document cover the installation of the Lynx Software Technologies, Inc. License Management software based on the FLEXnet Publisher (FLEXnet) version 11.14.1.3. This must be completed before it is possible to utilize any Lynx Software Technologies, Inc. tools or products protected by this version of FLEXnet Publisher. After installation of the FLEXnet Publisher (FLEXnet) software, the user must obtain and install a license key from Lynx Software Technologies, Inc..

If you are installing the Lynx License Management software on the same system where the previous version of the Lynx License Management software has been already installed, contact Lynx Software Technologies, Inc. technical support for the guidance on resolving compatibility issues.

Note that throughout the remainder of this document we will use the terms “FLEXnet Publisher” and “FLEXnet” interchangeably.

Currently the Lynx License Management software can be installed on the following systems:

- Red Hat® Enterprise Linux® 5 x86 or CentOS 5.x (32/64-bit)
- Red Hat Enterprise Linux 6 x86 or CentOS 6.x (32/64 bit)
- CentOS™ 7.x (32/64 bit)
- Windows® XP Professional
- Windows 7 Professional
- Solaris® (7 and higher)

NOTE: LynxSecure ships with only Linux® server binaries. If you need Windows or Solaris binaries and installation instructions, please contact Lynx Software Technologies, Inc. technical support.

Note that the FLEXnet server must be visible to ALL clients on the network which need to check out licenses.

FLEXnet Overview

FLEXnet is a client-server based licensing system that counts how many licenses are available and in use for a given licensed tool. A daemon called lmgrd runs on the license server and listens for license requests from protected products such as the Lynx Session Manager (`lwsmgr`). When a protected product starts, it checks a license file to find out where the license server is running. It then makes a request to the `lmgrd` daemon running on the server system or on the system where the product runs to check out a license. If a license is available, `lmgrd` issues it to the product, which can then continue to run normally. Periodically, `lwsmgr` must communicate with the `lmgrd` daemon to continue to run. When the product exits, the license is returned to the `lmgrd` daemon for someone else to use.

When started, the `lmgrd` daemon launches vendor daemons which are specified in the license file. The current version of the Lynx License Management software uses the `lynxrtsd` daemon. Each vendor daemon handles license requests for the corresponding vendor's products, so a single `lmgrd` daemon can handle license requests from different vendor's products.

The Lynx software products are licensed using one of two possible methods:

Node-Locked License

A single host/target version of the Lynx software, locked to run on a particular host computer. An example would be a single license to run the Red Hat Linux-hosted LynxSecure CDK on a particular Red Hat Linux system. Each system is uniquely identified by it's host ID that can be displayed by the `lmhostid` command or composite host ID that can be displayed using the `lwhostid` command. For example:

```
lwhostid = COMPOSITE=85B967ABE7D5
```

Floating License

A single host/target version of the Lynx software, able to run on any networked host of the same type. An example would be a single license to run the Red Hat-hosted LynxSecure CDK on any Red Hat machine on the network that could see the network license server. Floating licenses are bound to particular server machine by its host ID (`lmhostid`) or composite host ID (`lwhostid`) and allow the product to be shared by different development hosts.

The server machine that runs the license daemon must be a stable, networked machine running one of the systems specified in the Introduction section. The server does not need to run on the same machine as the development machine where you run a Lynx product such as Lynx Session Manager (`lwsmgr`). For example, you may install the FLEXnet license server on a central Linux host and run lwsmgr on any system which is supported by the Lynx product as a development host and which has a network connection with the FLEXnet server.

For node-locked licenses, the FLEXnet license server and Lynx products are normally installed on the same system and bound to its `lwhostid` or `lmhostid`. It is also possible to use different systems for the FlexNet license server and Lynx products.

For floating licenses, the FLEXnet license server and the Lynx cross-development hosts are normally separate systems.

After a server machine is chosen, the FLEXnet Publisher and Lynx License Management software is extracted from the CD-ROM. In the case of floating licenses, tools called `lwhostid` and `lmhostid` are run on the server to obtain the server's unique host ID. In the case of node-locked licenses, the `lwhostid` and `lmhostid` tool must be run on each host machine on which the Lynx products will run as well as the host on which the FLEXnet license server will run if this host is different from the cross-development host. These host IDs are then sent to Lynx by email or fax by filling the

Lynx Software Technologies License Key Request Form, and a key is then issued to the customer. Please refer to section "Obtaining a License" (page 4) for complete details on getting the license from Lynx. Once it is obtained, the key is added to the license file and is used by the Lynx software and the `lmgrd` server.

NOTE: The system used for running the FLEXnet software can be different from the development host.

Installing Lynx License Management Software on Linux Host

FLEXnet and Lynx License Management software for Linux host are distributed as a `Lynx-FlexLM` RPM package. The package installs the binaries into the `/opt/lw-flexlm/v11.14.1.3/linux64/bin` directory. This directory is added to the `PATH` environment variable when sourcing the `SETUP.bash` script (i.e. when setting up LynxSecure development environment).

If the license management utilities need to be used on a separate Linux host, the `Lynx-FlexLM` RPM needs to be installed from the LynxSecure installation media. Then, the directory with FLEXnet binaries needs to be added to `PATH` manually:

```
$ PATH=$PATH:/opt/lw-flexlm/v11.14.1.3/linux64/bin
```

Setting up the License Files with Dedicated License Server

In section "Setting up the License Files" (page 5), it was described how to set up the license file if the license server runs on the development host. If the development host(s) are different from the FLEXnet server host, use one of the methods described below to create the licenses for the development hosts.

Use a Local Copy of the License File.

Copy the license file from the FLEXnet server to a location of your choice on the development host. The license file can be edited on the development host to replace all lines except for the `SERVER` and `VENDOR` lines with the following line:

```
USE_SERVER
```

With this change, the license files on the development hosts will not need to be updated when the product license is renewed.

Use a Shared Copy of the License File Using NFS.

Mount the exported directory with the license file from the FLEXnet server, e.g.:

```
$ mount FLEXlm_server:/usr/local/flexlm /usr/local/flexlm
```

Adding LynxSecure License to an Existing License Management Server

If you already have the new Lynx License Management server running for products you received earlier from Lynx, add the lines from the new license except the SERVER and VENDOR lines to your existing license and execute the `lmreread` command.

If you already have an older version of the Lynx License Management server running for products you received earlier from Lynx Software Technologies, Inc., contact Lynx Software Technologies, Inc. technical support.

Using the Lynx License Management Software

Starting the FLEXnet License Management Server

Start the license manager daemon, `lmgrd`. The daemon `lmgrd` is the main daemon program for the FLEXnet distributed license management system. When invoked, it looks for the license file containing all required information about licensed products. The daemon should continue to run in the background.

On the Redhat Enterprise Linux host execute the following command:

```
$ lmgrd -c license_file_list
```

where *`license_file_list`* is the list of full pathnames to the license files that the server should utilize. If there are more than one license file the list items should be separated by a colon.

CAUTION! The host where the license server is to be running must have a host name which can be resolved to an IP address, using either DNS or the `/etc/hosts` file.

Another way to specify a license file to `lmgrd` is to set the `LM_LICENSE_FILE` environment variable prior to starting the server. For example:

```
$ export LM_LICENSE_FILE=license_file_list
$ lmgrd
```

The `LYNXRTSD_LICENSE_FILE` environment variable can also be used to specify the license file's absolute pathname or the list of absolute pathnames separated by a colon (on Windows, a semicolon). If both `LM_LICENSE_FILE` and `LYNXRTSD_LICENSE_FILE` are used, the latter will override the former.

FLEXnet Publisher environment variables can also be set in two other ways:

- In the process environment
- In the `$HOME/.flexlmrc` file

Check that the `$HOME/.flexlmrc` file is absent or does not contain the `LYNXRTSD_LICENSE_FILE` environment variable. If such file exists and contains environment variables that are needed for other vendor products, check that `LYNXRTSD_LICENSE_FILE` variable properly specifies the full pathname of your license file; otherwise, edit the value of the variable by adding the full pathname of your license file separated by a colon.

It is convenient to have the host operating system start the daemon automatically at boot time. Consult your operating system's user manuals for more information. For example, on Linux, you can do that from the following scripts:

- `/etc/rc.local`
- `/etc/rc[35].d/S*`

Checking the Status of FLEXnet Server

Check the installation using the `lmstat` command. Make sure that the `lmstat` command is in your `PATH` and that the `LM_LICENSE_FILE` environment variable is set before trying to run it. For example:

```
$ lmstat -a
```

The above command should produce output similar to the following:

```
lmstat - Copyright (c) 1989-2017 Flexera Software LLC. All Rights Reserved.
Flexible License Manager status on Fri 4/27/2018 20:08

License server status: 27000@licserver
```

```
    License file(s) on licserver: /usr/local/flexlm/flexlm14/license.dat:

licserver: license server UP v11.14.1

Vendor daemon status (on licserver):

  lynxrtsd: UP v11.14.1
```

For additional information about license server commands, refer to the FLEXnet Publisher documentation (*License Administration Guide*).

Stopping the License Management Server

To stop the License Management Server, use the `lmdown` command. For example:

```
$ lmdown
lmdown - Copyright (c) 1989-2017 Flexera Software LLC. All Rights Reserved.
Port@Host Vendors
1)  27000@licserver lynxrtsd
Are you sure (y/n)? y
15:24:37 (lmgrd) SHUTDOWN request from user2 at node licuser
15:24:37 (lmgrd) lmgrd will now shut down all the vendor daemons
15:24:37 (lynxrtsd) TCP_NODELAY NOT enabled
15:24:37 (lynxrtsd) Shutdown requested from user2@licuser IP=192.168.1.2
15:24:37 (lynxrtsd) daemon shutdown requested - shutting down
15:24:37 (lmgrd) Shut down FLEXnet lynxrtsd license server system on machine vald6
15:24:37 (lmgrd) EXITING DUE TO SIGNAL 15
1 FLEXnet License Server shut down
```

Working with the Lynx Session Manager

If the development host is the same as the FLEXnet server host, make sure that the `LM_LICENSE_FILE` environment variable is set as described above. If the development host is separate from the license server, set the `LM_LICENSE_FILE` environment variable to the full path of the development host's copy of the license, prepared using one of the methods described in section "Setting up the License Files" (page 5).

Then, change to the directory in which the Lynx CDK is installed and set up the environment. For example:

```
$ cd lynxsecure-installation-directory
$ . SETUP.bash
```

To start the Lynx Session Manager, enter the following command:

```
$ lwsmgr
Session Manager started
```

Once the Lynx Session Manager is started, the user can use the protected cross-development tools.

To stop the Lynx Session Manager, enter the following:

```
$ lwsmgr -s
Session Manager stopped
```

To check session information, enter the following:

```
$ lwsmgr
Current session status:
License type: floating license
Checkout date/time: Mon May 19 14:49:02 2014
Last auth date/time: Wed May 19 1910:44:24 2014
Expiration status: expires in 361 days
```

APPENDIX D

Changes in the Autoconfig Tool Command Line Interface

LynxSecure 6.0 Autoconfig Tool

In LynxSecure® 6.0, several global options have been retired in favor of subject-specific options. If the default option value needs to be overridden, it now needs to be done in each subject where a non-default value is to be used. Similarly, it is no longer possible to request creation of a certain virtual device in every subject; each subject must specify its full list of virtual devices explicitly.

Table D-1: Changes in Autoconfig Tool Commands

Deprecated Syntax (LynxSecure 5.3/LynxSecure 5.3.1)	New Syntax (LynxSecure 6.0)	Comments
`dump`	`export`	Renamed to a better name for DB management.
`load`	`import`	Renamed to a better name for DB management.
`merge`		The data base format has been reworked, so the command became unnecessary.

Table D-2: Changes in Autoconfig Tool Global Options

Deprecated Syntax (LynxSecure 5.3/5.3.1)	New Syntax (LynxSecure 6.0)	Comments
	`--auditsize`	New option sets the maximum number of entries in the LynxSecure audit buffer.
`--bios_image`		Moved to subject option `fwpath=FILENAME`.
`--cbit`	`--cbitpath`	Renamed for consistency with all options specifying filenames.
`--console`		Same as subject option `kernelconsole=TTY[:BAUD]`.
`--debug`	`-gg`	Replaced with a commonly used interface like in other UNIX tools.
	`--dtbpath`	The device tree blob path; currently Arm® specific option.
	`--flowpolicy`	New option to cover new requirements for HCV schema.
`--fvinitparams`		Same as subject option `initparams=LIST`.
`--fvs`		Same as subject option `fvspath=FILENAME`.
`--ibit`	`--ibitpath`	Renamed for consistency with all options specifying filenames.
	`--device-deps`	Specify the treatment of device dependencies when assigning devices to subjects.

Deprecated Syntax (LynxSecure 5.3/5.3.1)	New Syntax (LynxSecure 6.0)	Comments
`--initparams`		Same as subject option `initparams=LIST`.
`--info`	`-g`	Replaced with a commonly used interface like in other UNIX tools.
	`--keep-tmp-files`	Keep temporary files.
`--logger`		Same as subject option `logger`.
`--losbin`		Same as subject option `kdipath=FILENAME`.
`--lsabin`		Same as subject option `lsapath=FILENAME`.
`--memblocksize`		Removed.
`--monitor`		Same as subject option `monitor`.
`--policymaker`		Same as subject option `perm=readsched:writesched`.
`--ramdisk`		Same as subject option `ramdisk=FILENAME`.
`--ramsize`		Same as subject option `ram=SIZE`.
`--ropagesize`		Same as subject option `ropagesize=SIZE`.
`--skhheapsize`	`--skhheap`	New syntax allows for optional address specification of the SKH heap.
	`-o`	In previous releases, the optional last argument to the `mksrp` command was interpreted as the destination of where SRP was to be copied. This behavior is now controlled by the `-o` option.
`--sanbin`		Same as subject option `sanpath=FILENAME`.
`--serial`		Same as option `--diagoutput=serial`.
	`--show-device-deps`	Display device dependencies in the target hardware printout.
	`--show-hidden-devices`	Display devices disallowed for assignment in the target hardware printout.
`--vds_initrd`		Same as subject option `ramdisk=FILENAME`.
`--vds_ramdisk`		Same as subject option `ramdisk=FILENAME`.
`--vds_vmlinux`		Same as subject option `kernel=FILENAME`.
`--verbose`	`-ggg`	Replaced with a commonly used interface like in other UNIX tools.
`--virtaudio`		Same as subject option `virthda`/`virtac97`.
`--virtkma`		Same as subject option `virtkbdmouse,virthda`.
`--virtkvm`		Same as subject option `virtvga,virtkbdmouse`.
`--virtnet`		Same as subject option `virtnet`/`virte1000`/`virtne2000`.
`--virtusb`		Same as subject option `virtehci`/`virtohci`/`virtuhci`.
`--vmlinux`		Same as subject option `kernel`.
`--warnings`		All warnings are enabled by default.

Table D-3: Changes in Autoconfig Tool Subject-Specific Options

Deprecated Syntax (LynxSecure 5.3/LynxSecure 5.3.1)	New Syntax (LynxSecure 6.0)	Comments
`bdecdrom`	`virtcd`	Renamed for consistency with the rest of the virtual devices (all of them have prefix "virt").
`bdehdd`	`virthd`	Renamed for consistency with the rest of the virtual devices (all of them have prefix "virt").
`bderamdisk`	`virthd`	Renamed for consistency with the rest of the virtual devices (all of them have prefix "virt"). Additionally, the old option was superfluous, since if the user points to a file on the host machine, he wants ramdisk image.
`bootmenu`	`boottimeout`	Renamed as the current option name doesn't reflect the semantics of the option. It doesn't select any of the boot menus, but rather specifies the timeout before booting using the default boot device.
`cmdlocktimeout`		Not used.
`config`	`persistentstorage`	Renamed as the current option name is confusing.
`fvs32`	`fwpath`	Renamed for consistency with all options specifying filenames.
`fvs`	`fwpath`	Renamed for consistency with all options specifying filenames.
	`guestimage`	Option to specify the custom guest image required for the subject.
`hypercalls`	`perm=guesthypercall`	All permissions are grouped under `perm=` option.
`internetaccesstime`		Not used.
	`kernelconsole=TTY[:BAUD]`	Specifies the `console=` Linux® kernel boot option.
`kdiimg`	`kdipath`	Renamed for consistency with all options specifying filenames.
`kdijumpaddr`	`jump`	Renamed as the current name is long.
`kvm`		Removed; the option generated a lot of warning messages on the targets with no physical PS/2 keyboard or mouse. That raised a lot of questions during the training labs.
`linuxkernel`	`kernel`	Renamed for the sake of simplicity.
`linuxramdisk`	`ramdisk`	Renamed for the sake of simplicity.
`losbin`	`kdipath`	Renamed for consistency with all options specifying filenames.
`lsabin`	`lsapath`	Renamed for consistency with all options specifying filenames.
`mem`	`ram`	This option used to specify the subject RAM size; the new option name clarifies the usage more precisely.

Deprecated Syntax (LynxSecure 5.3/LynxSecure 5.3.1)	New Syntax (LynxSecure 6.0)	Comments
	`mode`	New option for setting the subject mode explcitily (e.g. for PV subjects with missing PV header in the boot image).
	`onfatalfault`	Added for the HCV fine tuning.
	`onwatchdog`	Added for the HCV fine tuning.
`oromimg`	`orompath`	Renamed for consistency with all options specifying filenames.
`pat`		Not used. All subjects have PAT on the x86 platform; this is not configurable anymore.
	`perm`	All permissions are grouped under this option.
`pcie`	`pciecfg`	Renamed to avoid option name collision with the physical device names on supported Arm boards.
`remotedisable`		Not used.
`sanbin`	`sanpath`	Renamed for consistency with all options specifying filenames.
`sanverbose`		Not used.
`shell`		Not used.
`sshauthentication`		Not used.
`tcpservices`		Not used.
`timeslice`		Reworked the default subject scheduling pocicy generation.
`udpservices`		Not used.
	`virtgic`	Adds virtual generic interrupt controller v2 to the subject. Arm only.
	`virttimer`	Adds virtual timer controller to the subject. Arm only.
`virtusb`		Removed; the user must explicitly specify all virtual USB controllers he wants in the particular subject.
`vmlinux`	`kernel`	Renamed for the sake of simplicity.
	`watchdogtimeout`	Added for the HCV fine tuning.
`openlid_action`		Removed.
`closelid_action`		Removed.
`acon_action`		Removed.
`acoff_action`		Removed.
`powerbutton_action`		Removed.
`sleepbutton_action`		Removed.
`batterylevel1`		Removed.
`batterylevel2`		Removed.
`batterylevel3`		Removed.

Deprecated Syntax (LynxSecure 5.3/LynxSecure 5.3.1)	New Syntax (LynxSecure 6.0)	Comments
`batterylevel1action`		Removed.
`batterylevel2action`		Removed.
`batterylevel3action`		Removed.

LynxSecure 5.3 Autoconfig Tool

The tables below summarize the changes in Autoconfig Tool command and option syntax.

Table D-4: Changes in Autoconfig Tool Commands

Deprecated Syntax (LynxSecure 5.2 and older)	New Syntax (LynxSecure 5.3/5.3.1)	Description
`addparts`	N/A	Starting with LynxSecure 5.3, it is recommended to use the Linux standard tools (`sgdisk`, `cgdisk`) to add a partition.
`diskinit`	N/A	Obsolete in LynxSecure 5.3, use the `sgdisk, cgdisk` .
`disklabel`	N/A	Obsolete in LynxSecure 5.3, use the `sgdisk, cgdisk` .
`indexrepo`	N/A	Obsolete in LynxSecure 5.3, as the VDS build process changed to buildroot.
`mkboot`	N/A	Obsolete in LynxSecure 5.3, as the SRP install supported by `install-srp` tool.
`nativeboot`	N/A	Obsolete in LynxSecure 5.3, as the ramdisk build process changed to buildroot.
`pvlinuxramdisk`	N/A	Obsolete in LynxSecure 5.3, as the ramdisk build process changed to buildroot.
`shrinkramdisk`	N/A	Obsolete in LynxSecure 5.3, as the ramdisk build process changed to buildroot.
`vdsinstall`	N/A	Starting with LynxSecure 5.3, the procedure for VDS installation to HDD is documented in section “VDS Root File System Customization and Booting” (page 74).
`vdsramdisk`	N/A	Obsolete in LynxSecure 5.3, as the ramdisk build process changed to buildroot.
`vdsramdiskshrink`	N/A	Obsolete in LynxSecure 5.3, as the ramdisk build process changed to buildroot.
`db disk`	N/A	Obsolete in LynxSecure 5.3, use `autoconfig db hardware *system-name*` to see whole hardware information.
`db fdisk`	N/A	Obsolete in LynxSecure 5.3, use Linux standard `sgdisk` and `cgdisk` commands.

Table D-5: Changes in Autoconfig Tool Global Options

Deprecated Syntax (5.2 and older)	New Syntax (LynxSecure 5.3/LynxSecure 5.3.1)	Description
`-aptchacher=`	N/A	Obsolete in LynxSecure 5.3, as the ramdisk build process changed to buildroot.
`-audio=`	`-virtaudio=`	Option renamed in LynxSecure 5.3.
`-autoconfigdist=`*`DISTRO-NAME`*	N/A	Obsolete in LynxSecure 5.3, as the ramdisk build process changed to buildroot.
N/A	`-bios_image=`*`FILENAME`*	New option in LynxSecure 5.3; supports user specified virtual BIOS image location.
`-bootdisk=`	N/A	Obsolete in LynxSecure 5.3, as now any partiton can be bootable and `install-srp` helps to install into any partition.
`-components=`	N/A	Obsolete in LynxSecure 5.3, as the ramdisk build process changed to buildroot.
`-componentscacher=`	N/A	Obsolete in LynxSecure 5.3, as the ramdisk build process changed to buildroot.
`-cpio`	N/A	Obsolete in LynxSecure 5.3, as the ramdisk build process changed to buildroot.
`-default-source=`*`URL`*	N/A	Obsolete in LynxSecure 5.3, as the ramdisk build process changed to buildroot.
`-disk=`*`DISK-NAME`*	N/A	Obsolete in LynxSecure 5.3, as disk partitions are created directly on the target from Hardware Discovery Linux instead of using the Autoconfig Tool.
`-dist=`*`DISTRO-NAME`*	N/A	Obsolete in LynxSecure 5.3, as the ramdisk build process changed to buildroot.
`-ext2`	N/A	Obsolete in LynxSecure 5.3, as the ramdisk build process changed to buildroot.
N/A	`-forcepolling=`*`DEVICE`*	New option in LynxSecure 5.3; supports user specified DEVICE interrupt polling.
`-fvsbin=`	`-fvspath=`	Option renamed in LynxSecure 5.3.
N/A	`-fvs32=`	New option in LynxSecure 5.3 to specify 32-bit FVS.
`-hotplug`	N/A	Obsolete in LynxSecure 5.3, use the block device persistence ID when assign disk partition to a subject.
N/A	`-info`	New option in LynxSecure 5.3 to print subjects resource allocation information during SRP creation
`-iso`	N/A	No longer needed in LynxSecure 5.3; ISO image is generated by default.

Deprecated Syntax (5.2 and older)	New Syntax (LynxSecure 5.3/LynxSecure 5.3.1)	Description
`-lsainitparams=`	N/A	Obsolete in LynxSecure 5.3, use subject's `initparams` option.
`-kma`	`-virtkma`	Option renamed in LynxSecure 5.3.
`-kvm=`	`-virtkvm`	Option renamed in LynxSecure 5.3.
`-kvmfocus=`	`-virtkvmfocus=`	Option renamed in LynxSecure 5.3.
`-lsk=`*`FILENAME`*	N/A	Obsolete in LynxSecure 5.3, hypervisor binary is generated on the fly by `mkcv`.
`-lsk32=`*`FILENAME`*	N/A	Obsolete in LynxSecure 5.3, hypervisor binary is generated on the fly by `mkcv`.
`-lskdebug`	N/A	Obsolete in LynxSecure 5.3, use the option `-runtimediag`
`-minimal`	N/A	Obsolete in LynxSecure 5.3, as the ramdisk build process changed to buildroot.
`-mkcvarg=`*`PARAMETERS`*	N/A	Obsolete in LynxSecure 5.3.
-realirq=*`DEVICE`*	`-nopolling=`*`DEVICE`*	Option renamed in LynxSecure 5.3.
`-ramdisk_dir=`*`DIR`*	N/A	Obsolete in LynxSecure 5.3, as the ramdisk build process changed to buildroot.
`-rif=`*`FILENAME`*	N/A	Obsolete in LynxSecure 5.3; RIF binary is generated on the fly by `mkcv`.
`-rif32=`*`FILENAME`*	N/A	Obsolete in LynxSecure 5.3, RIF binary is generated on the fly by `mkcv`.
N/A	`-runtimediag=`*`MODE`*	New option in LynxSecure 5.3 to enable run-time diagnostics.
N/A	`-savdebug=`*`DBGLEVEL`*	New option in LynxSecure 5.3 to enable FV subjects debug level. It is used for debug
`-scheduler=`*`ALGORITHM`*	N/A	Obsolete in LynxSecure 5.3; only the default scheduling algorithm is supported.
N/A	`-skhmodules=`*`MODULES`*	New option in LynxSecure 5.3 to enable LynxSecure Separation Kernel Hypervisor modules in the SRP.
N/A	`-smidisable`	New option in LynxSecure 5.3 to disable System Management Interrupts.
`-sskt=`*`VERSION`*	N/A	Obsolete in LynxSecure 5.3, not used anymore.
`-targetonly`	N/A	N/A
`-usbstick`	N/A	Obsolete in LynxSecure 5.3; `autoconfig.iso` can be installed to USB stick.
`-vds_initrd_amd64=`*`FILENAME`*	`-vds_initrd_x86_64=`*`FILENAME`*	Option renamed in LynxSecure 5.3.
`-vds_ramdisk_amd64=`*`FILENAME`*	`-vds_ramdisk_x86_64=`*`FILENAME`*	Option renamed in LynxSecure 5.3.
`-vds_vmlinux_amd64=`*`FILENAME`*	`-vds_vmlinux_x86_64=`*`FILENAME`*	Option renamed in LynxSecure 5.3.
N/A	`-virtusb`	New option in LynxSecure 5.3 to enable USB emulation in all FV subjects.

Table D-6: Changes in Autoconfig Tool Subject-Specific Options

Deprecated Syntax (5.2 and older)	New Syntax (LynxSecure 5.3/LynxSecure 5.3.1)	Description
`ac97`	`virtac97`	Option renamed in LynxSecure 5.3.
`virtdisk`	N/A	Option renamed in LynxSecure 5.3.
N/A	`bdetimeout=`*`TIMEOUT`*	New option in LynxSecure 5.3 to FV subject to specify BDE wait timeout.
`blockedcontexnt`	N/A	Obsolete in LynxSecure 5.3.
`blockedwebsites`	N/A	Obsolete in LynxSecure 5.3.
N/A	`bootmenu=`*`TIMEOUT`*	New option in LynxSecure 5.3 to FV subject to setup boot menu timeout.
`cdrom`	N/A	Obsolete in LynxSecure 5.3, assign the controller that CD-ROM is connected directly to the subject.
`config=`*`PARTTIONNAME`*	`config=`*`GPT-LABEL`*	Autoconfig-defined partition labels no longer supported in LynxSecure 5.3.
`consoleshell`	N/A	Obsolete in LynxSecure 5.3, no longer required to specify as only one variant of shell (`bash`) is supported.
`debug=on|off`	N/A	Obsolete in LynxSecure 5.3
`e1000`	`virte1000`	Option renamed in LynxSecure 5.3.
N/A	`hypercalls`	New option in LynxSecure 5.3 to allow FV subject to issue hypercalls.
`hda`	`virthda`	Option renamed in LynxSecure 5.3.
N/A	`initialstate=`*`[running|stopped]`*	New option in LynxSecure 5.3 to set FV subject's start state.
`kvmres=`*`[WxH|min|max]`*	N/A	Starting with LynxSecure 5.3 Autoconfig Tool automatically detects supported resolutions.
N/A	`linuxkernel`	New option in LynxSecure 5.3 to boot Linux RAM image.
N/A	`linuxramdisk`	New option in LynxSecure 5.3 to boot Linux RAM image.
N/A	`logger`	New option in LynxSecure 5.3 to enable SAV logging for FV Subjects
N/A	`monitor`	New option in LynxSecure 5.3 to enable monitor for FV Subjects.
`ne2000`	`virtne2000`	Option renamed in LynxSecure 5.3.
N/A	`pat`	New option in LynxSecure 5.3 to enable Page Attribute Table feature.
`patabde=`*`PARTITIONNAME`*	`bde=`*`PARTITIONNAME`*`,bdemode=legacy`	Syntax changed in LynxSecure 5.3.
`satabde=`*`PARTITIONNAME`*	`bde=`*`PARTITIONNAME`*`,bdemode=ide`	Syntax changed in LynxSecure 5.3.
`scvnc=`*`[true|false]`*	N/A	Obsolete in LynxSecure 5.3.
`subjectconnectivity`	N/A	Obsolete in LynxSecure 5.3.
N/A	`testpef`	New option in LynxSecure 5.3 to test FV subjects.
`usbany`	N/A	Obsolete in LynxSecure 5.3, as USB emulation rules syntax changed.

Deprecated Syntax (5.2 and older)	New Syntax (LynxSecure 5.3/LynxSecure 5.3.1)	Description
`vbiosdebug=`*`DBGLEVEL`*	N/A	Obsolete in LynxSecure 5.3.
N/A	`virtehci[:ndp=`*`NUM`*`]`	New option in LynxSecure 5.3 to enable USB EHCI emulation.
N/A	`virtkbdmouse`	New option in LynxSecure 5.3 to enable virtual keyboard and mouse to subject.
N/A	`virtohci[:ndp=`*`NUM`*`]`	New option in LynxSecure 5.3 to enable USB OHCI emulation.
`USBANY`, `USB`*x*`DEV`*y*, *`USB-device-ID`*	`virtuhci[:ndp=`*`NUM`*`]`	Syntax changed in LynxSecure 5.3 to support different types of USB host controllers (UHCI/OHCI/EHCI).
N/A	`virtvga`	New option in LynxSecure 5.3 to enable VGA emulation to a subject.
N/A	`wbinvd`	New option in LynxSecure 5.3 to specify handling of WBINVD CPU instruction.
`wirelessdisable`	N/A	Obsolete in LynxSecure 5.3, use Linux standard tools like `ifconfig` in VDS.
`wirelesscaninterval`	N/A	Obsolete in LynxSecure 5.3, use Linux standard tools like `iwlist` in VDS.
`xend=`*`[enable|disable]`*	N/A	Obsolete in LynxSecure 5.3.

APPENDIX E Glossary

Absolute Time
: The number of seconds since the epoch (Midnight, 1 January 1970 Universal Coordinated Time). Also referred to as Wall Time.

Authorized Subject
: has the ability to invoke privileged functions in the SKH Runtime Software. For example, an authorized subject can be given permission to request the SKH to startup or shutdown another subject. An authorized subject should be trusted by the system integrator to make proper use of its privileges, but is not trusted as part of SKH itself.

Autoconfig Tool
: A LynxSecure® command line utility to obtain target system hardware data and generate an HCV based on that data and the desired configuration passed as arguments.

Binary Configuration Vector
: A binary image of the unique configuration for a given SRP. Whenever the *Configuration Tool* is run, it produces an SRP which contains a Binary Configuration Vector.

Block Device Emulation (BDE)
: Provides virtual IDE/AHCI controller interfaces to a fully virtualized subject.

Bootloader
: When used in this generic sense, refers to the software that bootstraps the LynxSecure runtime software on a target platform.

Cache Allocation Technology (CAT)
: A feature available on Intel® processors. Allows the user to fine-tune the system performance by assigning portions of the last level cache (shared by all processor cores in a CPU package) to a particular task.

Capacity Bitmask (CBM)
: An integer number defining the amount of a system resource assigned to a particular task. The term is used in CPU resource partitioning technologies such as Intel RDT and CAT.

Certificate Authority (Certification Authority, CA)
: An entity in charge of issuing certificates to other entities. A root CA is the one implicitly trusted by the end-user software; the certificate issued by the root CA (root certificate) is self-signed. Certificates for intermediate CAs are signed by either the root CA or another intermediate CA.

Class of service (COS)
: An integer number identifying a CBM. The term is used in Intel CPU resource partitioning technologies such as RDT and CAT.

Configuration Tool
: A command line utility that converts a *Human Readable Configuration Vector* into the *Binary Configuration Vector*. It also enforces correct syntax and semantics in the data formats. Not to be confused with the *Autoconfig Tool*.

Configuration Vector
LynxSecure is configured using an offline configuration tool. The configuration vector is instantiated in multiple formats when in various stages of the system lifecycle: Human-Readable Configuration Vector (HCV), Binary Configuration Vector (BCV), and Runtime Configuration Vector (RCV).

Exported Resource
An exported resource is a resource made available by the Separation Kernel-Hypervisor (SKH) to one or more subjects.

External Interrupt
An interrupt generated by a real (non-virtual) device.

External Network
A network for which the LynxSecure platform is a node. LynxSecure interfaces with this network through a physical resource called a network interface card (NIC).

Full Virtualization
A mechanism by which an operating system, designed to run directly on a computing platform, can execute as a subject guest operating system without modification (i.e., without knowledge that it has been virtualized). Some implementations of full virtualization still use paravirtualized device drivers; but the operating system kernel itself remains unmodified.

Fully Virtualized Subject (FVS)
A component in LynxSecure responsible for platform initialization performed before executing the BIOS.

Guest Operating System (GuestOS)
An operating system executing in a virtualized context, rather than directly on the hardware with direct access to the hardware and privileged execution operations. When a GuestOS runs in LynxSecure it is also referred to as a Subject.
See Also Subject.

Human Readable Configuration Vector
An Extensible Markup Language (XML) file conformant to the XML Schema which defines LynxSecure configuration data.

Hypercall
An explicit software trap from subject code to the Hypervisor is a hypercall. A hypercall is similar to a system call on a non-virtual operating system environment. A hypercall allows indirect access to shared resources provided by the hypervisor.

Hypervisor or Virtual Machine Monitor (VMM)
For LynxSecure, software that runs directly on the hardware to provide virtualization for execution of multiple operating systems. In LynxSecure separation kernel and hypervisor are used interchangeably since both are realized at the same architectural layer.

Image
A contiguous collection of data and executable bits which makes up the LynxSecure on an executable hardware platform. The image is first deployed or downloaded as a file or a collection of files. When the hardware is powered on, the image is loaded from a storage medium, verified, and executed.

Installation and Packaging Toolset
The packaging toolset creates the runtime used to install LynxSecure onto the target platform. The installation tool is a runtime tool which installs LynxSecure onto a target system.

Installation Toolset
See Installation and Packaging Toolset.

Inter-Subject Communication Mechanisms

provided by the SKH allowing for subjects to communicate through a variety of well-defined interfaces. Each interface is provides different capabilities appropriate for different communication factors.

Interrupt

A system event, typically asynchronous, generated by a system device. An interrupt is generally used to indicate readiness of resources in an asynchronous manner. Interrupts usually have a well-defined source, path, and destination. Interrupts are usually "vectored," meaning that the control of some components of the source, path, and destination of a given interrupt are parameterized elements of an interrupt driven system.

Major Frame

A major frame consists of one or more minor frames. A major frame consists of a fixed amount of System Clock Tick (SCT) intervals. The sum of all SCT intervals assigned to each minor frame within a major frame equals the fixed amount of SCT intervals assigned to the major frame. A major frame is processor specific. Though two major frames may have identical minor frames, each major frame is distinct in that it is assigned to a unique processor and scheduling policy. In other words, a major frame is uniquely specified by its encompassing scheduling policy and assigned processor.

Memory region

A Memory region allows LynxSecure Security Kernel (LSK) to describe special handling required during the system boot up and during the working of the system.

Minor Frame

A minor frame consists of a subject id and a fixed number of SCT intervals. A minor frame binds a subject to a processor for a fixed amount of SCT intervals specified by the minor frame. Each minor frame belongs to a major frame.

Native

This is an operating system execution mode where the operating system has direct access to the hardware and privileged instructions without intermediary layers (such as a hypervisor or separation kernel). This is the typical way operating systems are deployed (no virtualization).

Paravirtualization

An approach of virtualization technology in which a guest OS is modified and recompiled prior to installation inside a virtual machine. A paravirtualized guest OS may run near native mode speed on the target hardware. This is also known as co-operative virtualization.

Platform Configuration Registers

A set of hardware registers provided by the Trusted Platform Module (TPM). These registers cannot be directly written to by software; instead, software *extends* a value into a register. The new value of the register is then a hash function of the old value and the extended value.

Processor Core

Modern processors may contain multiple processor cores. This is the smallest hardware processor resource that can be scheduled by the LynxSecure scheduling policy.

Read-Only Page (RO Page)

A per-subject memory structure describing the subject's configuration. Each subject can only access its own RO page, and the access is read-only. Despite the name, the structure is typically longer than one memory page. The RO page structure is defined in the `api.h` header file.

Reboot/Restart

Restarting a subject or the entire LynxSecure system under software control, without removing the power or (directly) triggering a reset line. It usually, though not always, refers to an orderly shutdown and restarting of the machine.

Resource

Resources are the totality of all hardware, firmware and software, and data that are executed, utilized, created, or protected by LynxSecure runtime software.

Resource Director Technology (RDT)

A feature available on Intel processors. Allows the user to fine-tune the system performance by monitoring and assigning a particular system resource (portion of the CPU cache, set the memory bandwidth, etc.) to a particular task.

RMRR

An RMRR memory region corresponds to a Reserved Memory Region lying in host physical memory. RMRRs are associated with devices that need a range of host physical memory to be identity-mapped in the DMA redirection tables for the device. These regions are typically used by USB and on-board graphics devices. RMRRs are not visible to the CPU; they are only accessible by device DMA. RMRRs may overlap with other memory region types, and RMRR mappings (memory flows) may overlap with mappings of other memory region types; in this case, the RMRR takes precedence over the other regions for DMA transfers to the overlapping address range. RMRR locations are determined by the host BIOS and reported to other software via ACPI tables.

Robustness (Medium Robustness and High Robustness)

A characterization of strength and assurance of security functions defined in Department of Defense Instruction 8500.2.

Scheduling Policy

A configuration encompassing a description of the subjects, and time frames involved and of how much time each subject executes upon all utilized processor cores on the target platform. A scheduling policy contains one major frame per processor core. A scheduling policy is the highest level of abstraction regarding scheduling. A scheduling policy consists of one or more major frames. Each major frame within a distinct scheduling policy is processor specific, meaning that each major frame is assigned to one and only one distinct processor. Only one scheduling policy is active at a time. A configuration vector can contain multiple scheduling policies.

Self-Assisted Virtualization (SAV)

A component in LynxSecure responsible for emulation of virtual devices. SAV executes in the context of the subject itself.

Separation Kernel

A software layer responsible for providing secure, non-bypassable isolation and separation of platform resources amongst entities wishing to utilize the platform resources. In LynxSecure, separation kernel and hypervisor are used interchangeably since both are realized by the same architectural component.

Separation Kernel / Hypervisor

A portion of the LynxSecure system responsible for providing separation kernel and virtualization (hypervisor-VMM). While providing different functionality, the terms are often combined due to their close intertwining into overlapping software layers within the overall architecture. Includes hardware and/or firmware and/or software mechanisms whose primary function is to establish, isolate and separate multiple *subjects* and control information flow between the subjects and exported resources allocated to those subjects.

SKH Runtime Package

A data format stored as a binary image which contains the runtime software and configuration to be loaded into memory by the *SKH* bootloader.

Subject

A subject is an active entity that causes operations to be performed, and executes within the context of the LynxSecure *Separation Kernel / Hypervisor*. Since the only information flows which are allowed are between subjects and exported resources, a subject is also considered a resource so that subject to subject information flows are possible.

Subject Execution State
A descriptive quality of the subject's current state in terms of its software execution, from the perspective of the SKH. Values include Running, Stopped and Suspended.

Synthetic Interrupt
Interrupts that are either generated internally by SKH and injected into a subject or initiated by a subject with right permissions and injected into another subject. The subject initiates the interrupt using a hypercall.

System Clock Tick
An event that causes each scheduler to be run in a near synchronous manner. SCTs occur at a periodic rate. The time between two SCTs is called a SCT interval. A SCT is an interrupt that is generated by a hardware timer. Also known as System Tick.

System Integrator
Develops code and configurations specific to a program or application to meet the needs of an end user.

System Timestamp Counter
A hardware dependent system counter, incrementing once for each specified hardware clock, that is started at, or soon after, power up or system reset, and which is not stopped or reset until the system is powered down or reset.

Target Board
The realized instance of a circuit board which executes the software.

Target Platform
The realized instance of a hardware platform which executes the LynxSecure software composed of a target board and many other hardware elements.

Timebase Calibration Value
A configuration value which describes length of one timebase cycle (i.e. the time between two subsequent timebase register increments) in nanoseconds. The timebase register is a platform register that increments at a constant speed. A system with a 2 GHz timebase register increment frequency would have a nominal Timebase Calibration Value of 0.5 nanoseconds. The subject that calibrates the system clock actually vary this value, speeding up and slowing down the apparent rate of time passage to synchronize the system to an external timebase in a software analogue of a phase locked loop.

Trusted Platform Module
A standard for a secure cryptoprocessor. LynxSecure bootloader and RIF can use the TPM device for reporting the metrics of the software modules, thus participating in the root of trust.

USB Emulation
Emulates USB controllers for fully virtualized subjects or provides a virtual USB controller for paravirtualized subjects.

Virtual KMA Switch
Provides support for switching the keyboard, mouse, audio and USB devices control between the fully virtualized subjects using key combinations.

Virtual KVM Switch
Provides support for switching the keyboard, mouse, video, audio and USB devices control between the fully virtualized subjects using key combinations.

Virtual Machine (VM)
The software that creates a virtualized environment upon which guest software executes. The following phrases are synonymous: "the Linux® and Windows® Guests," "the Linux and Windows guest OSs," and "the Linux and Windows virtual machines."

Virtual Network

An emulation of a network topology and node interfaces through software and wholly contained within the LynxSecure hardware platform itself. It uses well-known network protocols rather than custom network protocols. The virtual network can interface with an external network through software configuration and a physical NIC.

Virtualization

An added level of indirection and abstraction that hides physical characteristics of a resource from the way in which subjects use it. For LynxSecure, this allows multiple subjects (software) to utilize resources (hardware) where normally only a single subject could interface with the resource. This enables multiple operating systems to concurrently utilize a single hardware platform.

Wall Time

See *Absolute Time*.

Acronyms

Acronyms	Description
ABI	Application Binary Interface
ACPI	Advanced Control and Power Interface
AHCI	Advanced Host Controller Interface
API	Application Programming Interface
APIC	Advanced Programmable Interrupt Controller
BAR	Base Address Register
BCV	Binary Configuration Vector
BDE	Block Device Emulation
BIOS	Basic Input/Output System
BIT	Built-In Test
CA	Certificate Authority
CBIT	Continuously Running Built-In Test
CD	Compact Disc
CD-ROM	Compact Disc Read Only Memory
CPU	Central Processing Unit
DHCP	Dynamic Host Configuration Protocol
DMA	Direct Memory Access
DVD	Digital Versatile Disc
FLR	Function Level Reset
FVS	Fully Virtualized Subject (LynxSecure component)
GPA	Guest Physical Address
HCV	Human Readable Configuration Vector
HPA	Host Physical Address
I/O	Input/Output
IBIT	Initiated by an Event Built-in Test
I/O MMU	Input Output Memory Management Unit
IP	Internet Protocol
IPI	Inter Processor Interrupt

Acronyms	Description
IRQ	Interrupt Request
KMA	Keyboard Mouse Audio
KVM	Keyboard Video Mouse
LPC	Low Pin Count (bus)
LSA	LynxSecure Application
LSK	LynxSecure Separation Kernel
MAC	Media Access Control
MSI	Message Signaled Interrupts
NAT	Network Address Translation
NIC	Network Interface Card
OS	Operating System
PCR	Platform Configuration Register
PCI	Peripheral Component Interconnect
PCIe	PCI Express
PSCI	Power State Coordination Interface
PXE	Preboot Execution Environment
RIF	Runtime Initialization Function
RMRR	Reserved Memory Range Region
SATA	Serial Advanced Technology Attachment
SAV	Self-Assisted Virtualization (LynxSecure component)
SBIT	Startup Initiated Built-In Test
SCT	System Clock Tick
SESM	Subject Execution State Manager
SKH	Separation Kernel Hypervisor
SMP	Symmetric Multiprocessing
SMT	Symmetric Multithreading
SR-IOV	Single Root I/O Virtualization
SRP	SKH Runtime Package
SW	Software
TCP	Transmission Control Protocol
TSC	Time Stamp Counter
USB	Universal Serial Bus
VDS	Virtual Device Server
VGA	Video Graphics Array
VF	(SR-IOV) Virtual Function
VM	Virtual Machine
VT	Virtualization Technology
VT-d	Intel's VT for devices (includes I/O MMU)
XML	Extensible Markup Language