CHARON-VAX V4.8 for Linux Users Guide

Contents

Introduction

Table of Contents

General Description

VAX Hardware Virtualization allows users of HP VAX computers to move application software and user data to a non-VAX platform without having to make changes to software and data. VAX Hardware Virtualization is a software solution that replaces VAX hardware.
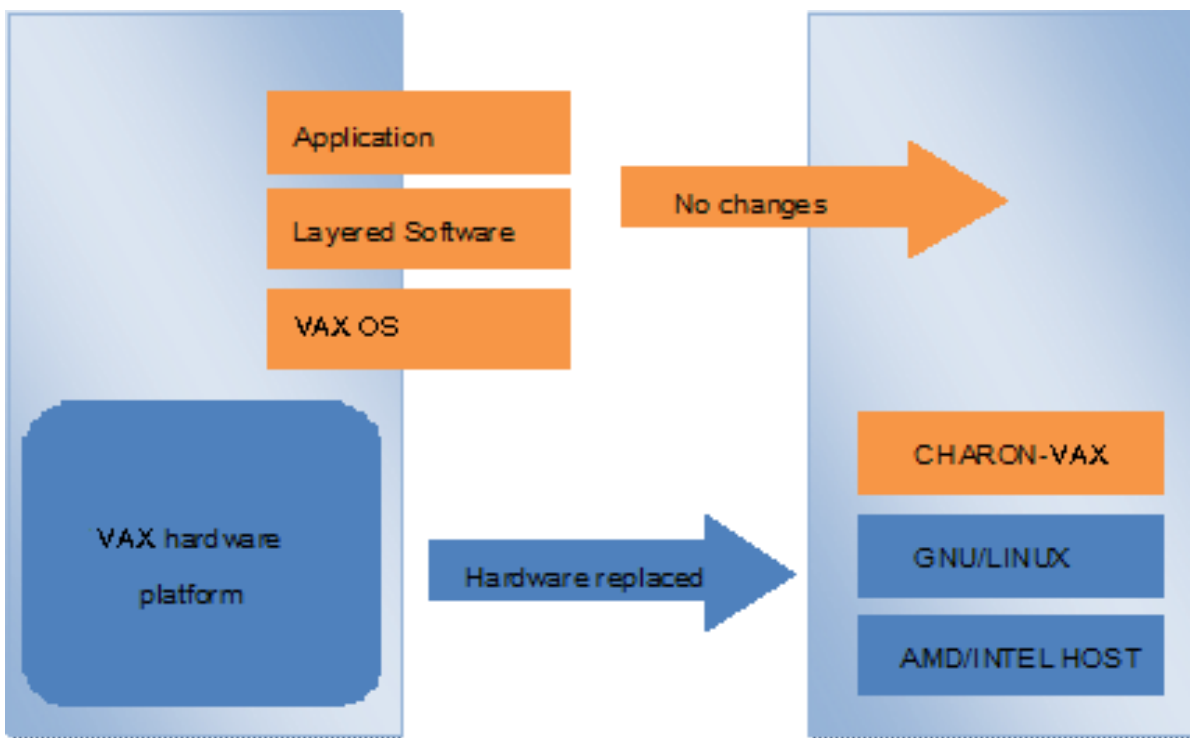
This approach is best understood when the VAX Hardware Virtualization Software is viewed as a special interface between the old VAX software and a new hardware platform. Basically, the CHARON software presents a VAX hardware interface to the original VAX software, so that the existing software cannot detect a difference. This means no changes have to be made to the existing software. User programs and data can be copied to a new modern industry standard server (64-bit or 32-bit Intel or AMD) and continue to run for many more years.

The VAX virtualization software is designed to replace single and multi-CPU VAX computer systems, including:

- MicroVAX II
- MicroVAX 3600
- MicroVAX 3900
- MicroVAX 3100 models 96 and 98
- VAXserver 3600
- VAXserver 3900
- VAXstation 4000 models 90, 106, 108, 700 and 705
- VAX 6310
- VAX 6610, 6620, 6630, 6640, 6650 and 6660

The principles of VAX Hardware Virtualization

In order to make the correct decisions regarding how to apply VAX Hardware Virtualization or Emulation, it is important to make a distinction between the VAX hardware that is virtualized or emulated and the (non-VAX) hardware from the (non-VAX) hardware host platform that carries the VAX Virtualization Software.

Virtualized hardware

CHARON-VAX virtualizes various VAX architectures and meets or exceeds the performance level of these VAX systems when run on the recommended hardware platform. Our VAX emulator product currently contains the following VAX models:

- MicroVAX II
- MicroVAX 3600
- MicroVAX 3900
- VAXserver 3600 (depending on license supports up to 512 MB of emulated memory)
- VAXserver 3900 (depending on license supports up to 512 MB of emulated memory)
- MicroVAX 3100 models 96 and 98
- VAX 4000 models 106, 108, 700 and 705
- VAXstation 4000 model 90
- VAX 6310
- VAX 6610
- VAX 6620
- VAX 6630
- VAX 6640
- VAX 6500
- VAX 6660

The following table explains which hardware boards we virtualize:

Subsystem	Covered VAX hardware
Serial Lines Controllers	UART, QUART, CXA16, CXB16, CXY08, DHQ11, DHV11, DZV11, DZQ11, DL11, DLV11, DZ11, DHW42-AA, DHW42-BA, DHW42-CA
QBUS Disks/Tapes Controllers	RQDX3, KDB50, TQK50, TUK50, KDM70
SCSI Controllers	NCR53C94
DSSI Subsystem	SHAC, HSD50
CI Subsystem	CIXCD, HSJ50
Network Controllers	DEQNA, DESQA, DELQA, DEUNA, DELUA, DEMNA, DEBNI, PMADAA

Host platform

The Virtualization Software presents standard VAX devices to the VAX operating system, allowing the OS to function as though it were still running on a VAX computer. For example, virtual disk container files in a directory or device files in /dev directory of the host Linux platform are presented by the Virtualization Software to the VAX OS as emulated SCSI disks attached to a SCSI adapter.
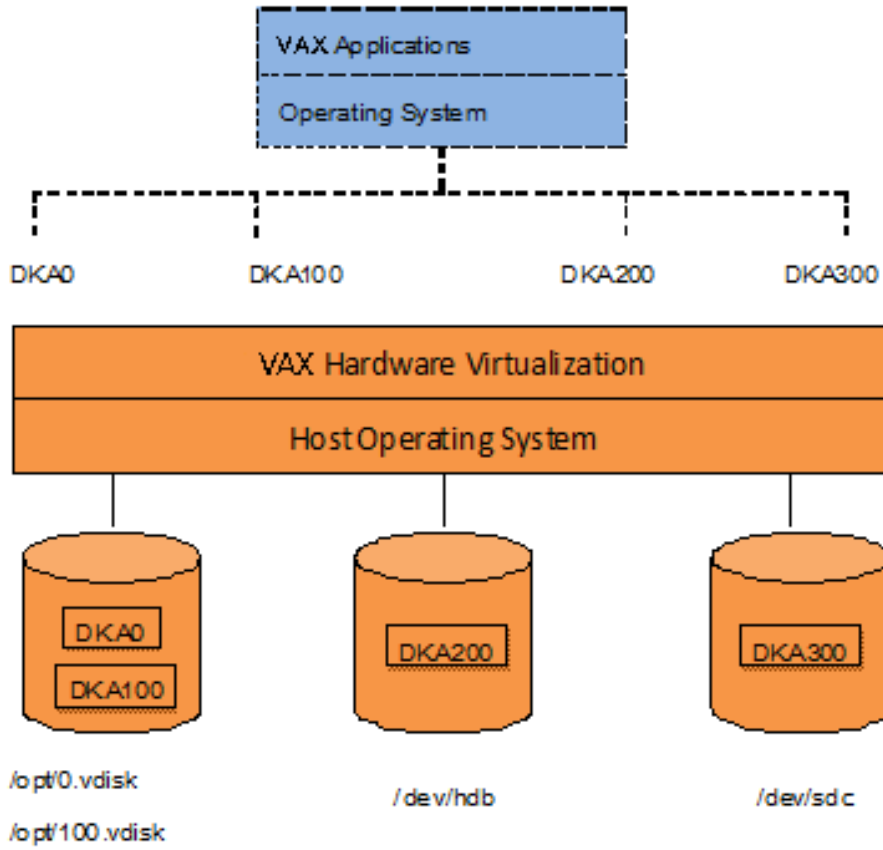
With the use of current storage technology, disks do not have to be physically attached to the Host platform, they can also reside on a SAN or iSCSI storage structure.

A similar translation process is also valid for other emulated hardware devices.

Back to Table of Contents

Conventions

Throughout the document(s) these conventions are followed

Notation	Description
$	The dollar sign in interactive examples indicates an operating system prompt for VMS. The dollar sign can also indicate non superuser prompt for UNIX / Linux.
#	The number sign represents the superuser prompt for UNIX / Linux.
>	The right angle bracket in interactive examples indicates an operating system prompt for Windows command (cmd.exe).
User input	Bold monospace type in interactive examples indicates typed user input.
\<path\>	Bold monospace type enclosed by angle brackets indicates command parameters and parameter values.
Output	Monospace type in interactive examples, indicates command response output.
[ ]	In syntax definitions, brackets indicate items that are optional.
...	In syntax definitions, a horizontal ellipsis indicates that the preceding item can be repeated one or more times.
dsk0	Italic monospace type, in interactive examples, indicates typed context dependent user input.

The following definitions apply

Term	Description
Host	The system on which the emulator runs, also called the Charon server
Guest	The operating system running on a Charon instance, for example, Tru64 UNIX, OpenVMS, Solaris or MPE

CHARON-VAX for Linux installation

Table of contents

Introduction

CHARON-VAX products are distributed in form of archive TAR.GZ files that contain RPM modules for different components. Generally it is recommended to install all the RPM modules but it is possible to omit some RPM files if they are not needed.

CHARON installation consists of the following steps:

- Host system checks (hardware and software) to ensure the host platform meets the minimum CHARON-VAX installation requirements
- Installation of any 3rd party material, for example, the utilities required for CHARON-VAX
- Extracting CHARON-VAX RPM modules from the TAR.GZ archive and their individual installation
- Installation of the CHARON-VAX license (hardware dongle or software license)
- Configuration of the CHARON-VAX host system. It assumes creating a specific user, configuring the network, etc.

Back to Table of Contents

Hardware Requirements

Number of CPU cores

Each CHARON emulated CPU requires a corresponding physical core. So the total number of the host CPUs must exceed the number of emulated CPUs since some of the host CPUs must be dedicated to serving CHARON I/O operations and host operating system needs. If several CHARON instances run in parallel, the required number of CPU cores is cumulative.

The table below lists the minimum and recommended number of CPUs required for each product:

CHARON-VAX model	Minimal number of CPU cores	Recommended number of CPU cores
VAX 6610	2	4
VAX 6620	3	4
VAX 6630	4	6
VAX 6640	6	8
VAX 6650	8	12
VAX 6660	8	12
Other models	2	2

Hyper-threading must be switched off completely. Disable hyper-threading in the BIOS settings of the physical host or, for a VMware virtual machine, edit the virtual machine properties, select the Resources tab then select Advanced CPU. Set the Hyper-threaded Core Sharing mode to *None*.

CPU type and speed

Since CHARON-VAX utilizes LAHF instruction in VAX CPU emulation please avoid usage of early AMD64 and Intel 64 CPUs in CHARON host system since they lack it. AMD introduced the instruction with their Athlon 64, Opteron and Turion 64 revision D processors in March 2005 and Intel introduced it with the Pentium 4 G1 stepping in December 2005.

Concerning CPU speed, the general recommendation is that higher the CPU frequency is, better the emulated VAX performances will be. The minimum recommendation is at least 3 GHz.

Operative memory

The minimum host memory size:

- depends on the amount of VAX memory to be emulated and on the number of CHARON-VAX instances to be executed on one host.
- is calculated according to the following formula:

> The minimum host memory = (2Gb + the amount of VAX memory emulated) per CHARON-VAX instance.

The maximum amount of VAX memory that can be created in the CHARON-VAX/66x0 products and supported by OpenVMS/VAX is *3584* Mb. For details, see the memory size specifications.

Disk storage

The total amount of disk space required for CHARON-VAX can be calculated as a sum of all the disk/tape image sizes plus 50 MB for the CHARON software plus the space required for the host operating system. Temporary disk storage is often needed when setting up a new virtual machine (for source disks backups storage, software installation kits, etc...).

When virtual disks/tapes are used to represent physical disk drives / magnetic tapes, the disk/tape image files have the same size as their hardware equivalent, regardless of their degree of utilization.

Ethernet adapters

CHARON-VAX networking requires dedicated host Ethernet adapters; their number must be equal to the emulated adapters to be configured in CHARON-VAX. One adapter (optionally) can be left to the host for TCP/IP networking, management interface, etc.

It is also possible to use virtual network interfaces but for performance considerations, it is recommended to use physical ones only.

For VMware based CHARON hosts, it is mandatory to use "E1000" virtual network adapters. "E1000E" adapters are not supported.

Software Requirements

- Red Hat Enterprise Linux 7.x, 64bit
- Red Hat Enterprise Linux 6.5 - 6.8, 64bit
- CentOS 7.x, 64bit
- VMware ESXi 5.5 and 6.x up to 6.5 (requires a supported Linux operating system on top of a ESXi virtual machine)

> For CentOS, a connection to the internet is required to install the `glibc.i686` package which is not included in the Standard distribution DVD. If there's no connection available, please use the "Everything" distribution DVD.

Host system preparation

> The automatic installation of updates must be disabled. Updates to the CHARON host must be done only in specific service maintenance periods established by the system administrator. Before applying new updates one must shutdown the operating system running on CHARON and stop all the running CHARON instances and services.

If a network-wide license (red dongle or software license) is going to be used, do the following:

- *On the server side (where the network license will reside)*: open port 1947 for both TCP and UDP
- *On the client side*, if broadcast search for remote licenses is to be used, UDP traffic from port 1947 of the license server to ports 30000-65535 of the client must be permitted.
- *Both on server and client sides*: set default gateway

Please consult with your Linux User's Guide on details.

> If stricter firewall rules are required, it is possible to open the ports 30000-65535 and 1947 only for the "`/usr/sbin/hasplmd`" daemon.

Before installation

1. Login as the superuser ("*root*") on the host system. Because Sentinel HASP runtime relies on 32-bit compatibility libraries to run on Linux, the 32-bit compatibility libraries must be installed before continuing. If the emulator host has access to a package repository, either local or remote, use the following command:

```
# yum install glibc.i686
```

> ℹ️ Sometimes it is not possible to use an online repository for the installation of 32-bit glibc package. In this case the procedure described in the appendixes has to be used: glibc.i686 installation without Internet connection

2. Create a directory for the CHARON-VAX distribution, copy the TAR.GZ files there and change to this directory as shown in the following example:

```
# mkdir /charon_dist
# cp /tmp/charon-vax-4.8-18302.el71.tar.gz /charon_dist
# cd /charon_dist
```

> **WARNING**
> - If you plan to install CHARON-AXP on the same server, both products, CHARON-AXP and CHARON-VAX, will have to be the same build number.
> - If you upgrade from a previous version of CHARON-VAX, please stop all running CHARON virtual machines and uninstall CHARON products before proceeding with the installation steps described below.

Distribution preparation

1. Extract the content of the distribution TAR.GZ files to the current directory:

```
# tar -xvzf charon-vax-<VER>-<BN>.<ZZ>.tar.gz
```

where:

	Description
VER	Version of CHARON-VAX product, for example 4.8
BN	Build Number of CHARON-VAX product, for example 18302
ZZ	CHARON-VAX target operating system identifier. For CentOS 7.x '*ZZ*' value is '*el71*', for Red Hat Enterprise Linux 7.x the value is '*el71*' and for Red Hat Enterprise Linux 6.5 to 6.8 the value is '*el65*'.

Example:

```
# tar -xvzf charon-vax-4.8-18302.el71.tar.gz
```

As result, a new directory "*charon-vax-<VER>-<BN>.<VC>.<ZZ>*" will be created.

2. Switch to that directory:

```
# cd charon-vax-<VER>-<BN>.<ZZ>
```

Example:

```
# cd charon-vax-4.8-18302.el71
```

3. The distribution directory contains the following RPM files:

File name	Description
charon-vax-VER-BN.ZZ.x86_64.rpm	CHARON-VAX
aksusbd-2.5-1.i386.rpm	HASP Run-time
charon-license-VER-BN.ZZ.x86_64.rpm	CHARON Libraries
charon-utils-VER-BN.ZZ.x86_64.rpm	CHARON Utilities

Example:

```
# ls
aksusbd-2.5-1.i386.rpm
charon-vax-4.8-18302.el71.x86_64.rpm
charon-license-4.8-18302.el71.x86_64.rpm
charon-utils-4.8-18302.el71.x86_64.rpm
```

Installation

Issue the following command to install all the RPM files present in the directory:

```
# yum install *.rpm
```

Enter "**y**" to agree to install all the listed packages.

Example:

```
Dependencies Resolved

================================================================================
Package Arch Version Repository Size
================================================================================
Installing:
aksusbd i386 2.5-1 /aksusbd-2.5-1.i386 2.9 M
charon-vax x86_64 4.8-18302 /charon-vax-4.8-18302.68704.el71.x86_64 260 M
charon-license
x86_64 4.8-18302 /charon-license-4.8-18302.68704.el71.x86_64 2.9 M
charon-utils
x86_64 4.8-18302 /charon-utils-4.8-18302.68704.el71.x86_64 1.8 M

Transaction Summary
================================================================================
Install 4 Packages

Total size: 267 M
Installed size: 267 M
Is this ok [y/d/N]:y
```

Check the installation process has completed successfully.

Example:

```
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction (shutdown inhibited)
Installing : aksusbd-2.5-1.i386 1/4
Starting aksusbd (via systemctl): [ OK ]
Installing : charon-utils-4.8-18302.x86_64 2/4
Installing : charon-license-4.8-18302.x86_64 3/4
Installing : charon-vax-4.8-18302.x86_64 4/4
Verifying : aksusbd-2.5-1.i386 1/4
Verifying : charon-license-4.8-18302.x86_64 2/4
Verifying : charon-vax-4.8-18302.x86_64 3/4
Verifying : charon-utils-4.8-18302.x86_64 4/4

Installed:
aksusbd.i386 0:2.5-1 charon-vax.x86_64 0:4.8-18302
charon-license.x86_64 0:4.8-18302 charon-utils.x86_64 0:4.8-18302

Complete!
```

Re-login (as "*root*") to apply the PATH settings or execute the following command:

```
# . /etc/profile.d/charon.sh
```

> Note that the "charon-utils" package has the following dependencies:
>
> - ethtool
> - bridge-utils
> - net-tools
> - iproute
> - NetworkManager
>
> During "ncu" installation using "yum", these packages will be installed automatically if some of them are absent on the host system.

CHARON-VAX home directory

By default CHARON is installed in the "`/opt/charon`" directory. It has the following subdirectories:

Directory	Description
/bin	Contains all the executable files
/cfg	Contains the configuration files templates
/doc	Contains the documentation
/log	Contains the log files
/disks	Contains the disk containers
/drivers	Contains the CHARON drivers

The most important at this stage is the "/cfg" directory since it contains template configuration files with examples of typical configuration parameters and comments. We will focus our attention on this subject in the next chapter.

Specific user account creation

Create a specific user account named "*charon*" for running CHARON:

```
# useradd -G disk,tape,cdrom,dialout,lock -c "Charon User" -m charon
# passwd charon
```

Any existing user can also be used to run CHARON. In this case issue the following command to include this existing user into these specific groups:

```
# usermod -G disk,tape,cdrom,dialout,lock -g <user name> <user name>
```

Example:

```
# usermod -G disk,tape,cdrom,dialout,lock -g tommy tommy
```

> The specific account created above does not allow to use physical consoles "`/dev/tty<N>`" as CHARON consoles. If you plan to map CHARON console to "`/dev/tty<N>`" use only "root" account for CHARON running.

License installation

Regular HASP USB dongle

If CHARON license is located on a regular USB dongle, just connect it to the host USB port.

> If the CHARON host is accessed remotely, please note that regular HASP licenses cannot be displayed and used to start a CHARON virtual machine. As a workaround it is possible to install CHARON as a daemon (service). This procedure will be described later.

Network HASP USB dongle

If the CHARON license is a network license (red USB dongle), it is possible either to connect it to the host USB port (to use it locally and provide it to other hosts on the local network at the same time) or to install it on a local network "license server" for remote access from this particular host.

If a remote license server is to be used:

- Copy the aksusbd-2.5-1.i386.rpm and charon-license-4.8-<*build*>.<*OS identifier*>.x86_64.rpm files (see above) to the server, for example "/tmp"
- Login as "*root*" to the server.
- Switch to that directory.
- Install the copied files using "yum".
  Example:

```
# cd /tmp
# yum install aksusbd* charon-license-*
```

- Connect the network HASP dongle to one of the server USB ports.

> The network HASP (red dongles) licenses have no restrictions with respect remote access

Software license

If CHARON license is a software license (SL), it is installed on the host using the following procedure:

1. Run hasp_srm_view utility in the following way to get the host fingerprint file ("my_host.c2v" in this example):

```
# hasp_srm_view -fgp my_host.c2v
```

2. Send the resulting file to STROMASYS. In return STROMASYS will provide you with a ".v2c" file, for example "your_license.v2c".
3. Copy the received file to any folder on the CHARON host, invoke the system default web browser and enter URL http://localhost:1947 to display the "**Sentinel Admin Control Center**" (**ACC**) web interface. This interface allows you to view and manage the CHARON licenses.
4. In the **ACC** perform the following steps: select **Update/Attach** from the menu on the left pane then use the **Browse** button to select the received file and click on the **Apply File** button to install the license.
5. Ensure that the software license is now visible in the "**Sentinel Keys**" section of the **ACC**.

> It is also possible to use the "hasp_update" utility for applying ".v2c" files.

> The Software Licenses (SL) are always network licenses, they have no restrictions with respect to being displayed or accessed via a remote connection.

> A "Provisional" (demo) license does not require collecting a fingerprint. For its installation start at step 3 in the sequence above

License validity verification

To check the CHARON license validity, invoke the hasp_srm_view utility to make sure that CHARON license is visible is correct:

- Text of the license is displayed correctly by the hasp_srm_view utility, no error messages are shown.
- The content of the license looks correct. For example: license number, major and minor versions, minimum and maximum build numbers, CHARON-VAX products and allowed hardware (CHARON-VAX models) should be checked. More details on the license content can be found in the CHARON-VAX Licensing chapter of this Guide.

Example:

```
# hasp_srm_view

License Manager running at host: dlao.msc.masq
License Manager IP address: 192.168.1.129

HASP Net key detected

The Physical KeyId: 1422726238
CHARON Sentinel HASP License key section
Reading 4032 bytes

The License Number: 000.msc.sanity.tests
The License KeyId: 1422726238
The Master KeyId: 827774524
Release date: 12-SEP-2016
Release time: 15:15:15
Update number: 1
End User name: MSC
Purchasing Customer name: STROMASYS
...
```

If multiple licenses are available, it is possible to check them using the "-all" parameter with the hasp_srm_view utility in the following way:

```
# hasp_srm_view -all
```

It it also possible to display the license content for one specific key using the "-key" parameter and specifying the Key Id (see "# **hasp_srm _view -h**" for more)

Reminder: If the CHARON host is accessed over a remote connection, please note that regular HASP licenses cannot be displayed and used in this case. As a workaround it is possible to install CHARON as a daemon (service). This procedure will be described later.

Troubleshooting

If the CHARON license content cannot be displayed by the hasp_srm_view utility or it is incorrect, check the license is available and correctly used:

1. Invoke the system default web browser and enter the URL http://localhost:1947 to display the "**Sentinel Admin Control Center**" (**ACC**) web interface.
2. Click on "**Sentinel Keys**" link to open the corresponding page.
3. Make sure that one and only one CHARON HASP or SL license is present.

Problem	Action
No license is displayed	Make sure that all the recommendations above about remote access to the host are fulfilled (if remote access takes place), that the HASP USB key is not broken and its LED indicator is lit (meaning that it is used by the host).
Only one License key / SL is seen and its content is incorrect	Contact STROMASYS to request a new license update.
Several License keys / SLs are displayed	Remove all of them except the one provided by STROMASYS for the just installed version of CHARON.

Removing licenses can be done by physical disconnection of the corresponding USB HASP keys from CHARON host and physical disconnection of the network HASP keys from all hosts on the local network (or by disabling remote access to network licenses from the CHARON host - see detailed explanations below).

It is also possible to disable access to network licenses if only a local license is to be used: Click on the "**Configuration**" link to open the "**Configuration for Sentinel Manager**" page.

Uncheck "**Allow Access to Remote Licenses**" and "**Broadcast Search for Remote Licenses**" check boxes from the "**Access to Remote License Managers**" tab then press the "**Submit**" button to apply changes.

It is also possible to leave several licenses available to CHARON-VAX at the same time but in this case they have to be specified in the configuration file.

Example:

```
set session license_key_id[0]=1877752571
```

It is also possible to have one "main" and one "backup" license in case the main license becomes unavailable:

```
set session license_key_id[0]=1877752571 license_key_id[1]=354850588
```

CHARON-VAX checks its licenses from time to time starting with main license. If it becomes unavailable, it attempts to access the backup license.

Network configuration

In most cases CHARON will use a network. In this case CHARON requires one or more dedicated network interfaces with any other protocols including TCP/IP removed at the host level.

Two ways of network configuration are possible:

- With the help of the "ncu" utility
- Manual

The first way is recommended. Use the manual approach only in absence of the "ncu" utility or if it is impossible to use it.

Configuration with NCU utility

Login as root and enter "ncu". The following menu will appear:

```
# ncu

CHARON Network Configuration Utility, STROMASYS (c) 2016 Version 1.6

Interfaces Dedicated to State
---------- ------------ ------------
eth0       host         connected to host
eth1       host         disconnected from host

==============================VLAN==============================

===============================================================

select action:
1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit
:> 1
```

The utility lists the available network interfaces (both physical and virtual) and indicates whether they are dedicated to the host or to CHARON and whether they are currently in use by the host operating system.

"ncu" offers several options:

- Dedicate interface to CHARON (option "1")
- Release interface to host (option "2")
- Create a bridge between a chosen physical network interface and the Linux virtual network and create a number of virtual network interfaces (option "3")
- Remove the Linux virtual network and all the created virtual network interfaces (option "4")
- Add VLAN (option "5")
- Remove VLAN (option "6")
- Print status (option "7") - use it to display status of network interfaces and the menu shown above
- Exit (option "8")

In the example above we see 2 network interfaces, "eth0" and "eth1", that are are dedicated to the host and the host uses only the interface "eth0".

Let's dedicate the interface "eth1" to CHARON-VAX.

Enter "1" then "eth1":

```
Specify the interface to dedicate to CHARON:eth1

Turning off offloading for eth1.. Please wait

select action:
1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit

:> 7
```

Now the interface "eth1" is dedicated to CHARON-VAX:

```
Interfaces Dedicated to State
---------- ------------ ------------
eth0       host         connected to host
eth1       CHARON       disconnected from host

================================VLAN================================

====================================================================

select action:
1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit

:>
```

Enter "8" to return to the console prompt.

Now "eth1" can be used by CHARON-VAX.

Manual Configuration

Choosing network interface

To choose an interface to be used for CHARON networking, do the following:

```
# ifconfig
eth0 Link encap:Ethernet HWaddr 00:60:52:0A:A9:1E
...
eth1 Link encap:Ethernet HWaddr 00:C0:26:60:FB:15
...
eth2 Link encap:Ethernet HWaddr 00:1A:92:E1:3F:7F
```

Choose an interface to be used by CHARON, for example "*eth1*"

Designation of network interface to CHARON

To designate the chosen interface to CHARON open up the file "`/etc/sysconfig/network-scripts/ifcfg-eth`*N*" (where *N* is the number of the interface to be used for CHARON, in this case it is "1") and make sure that all the IP-setup related parameters are removed. The file must look like this:

```
DEVICE="eth1"
HWADDR="00:06:2B:00:6A:87"
NM_CONTROLLED="no"
ONBOOT="no"
```

Switching off the offload parameters

Determine what additional parameters are currently set to "*on*" on the host network adapter to be used by CHARON using the following command:

```
# ethtool -k <device>
```

Example:

```
# ethtool -k eth1
Offload parameters for eth1:
rx-checksumming: on
tx-checksumming: on
scatter-gather: on
tcp-segmentation-offload: off
udp-fragmentation-offload: off
generic-segmentation-offload: on
generic-receive-offload: off
large-receive-offload: off
```

Use "ethtool" to switch off all the offload parameters:

```
# ethtool -K <device> <parameter> off
```

Example:

```
# ethtool -k eth1
Offload parameters for eth1:
rx-checksumming: on
tx-checksumming: on
scatter-gather: on
tcp-segmentation-offload: off
udp-fragmentation-offload: off
generic-segmentation-offload: on
generic-receive-offload: off
large-receive-offload: off
```

For the example above let's create a temporary file containing the commands to be executed at system startup as the offload parameters must be switched off following each reboot:

```
ethtool -K eth1 rx off
ethtool -K eth1 tx off
ethtool -K eth1 sg off
ethtool -K eth1 gso off
ethtool -K eth1 gro off
```

Let's suppose the name of the file is "`offload_off_eth1.txt`". To execute it on system startup, execute the following command (example):

```
# cat offload_off_eth1.txt >> /etc/rc.d/rc.local
```

[Back to Table of Contents](#)

Final steps

- Reboot the host system
- Login as user "*charon*"
- Verify the offload parameters are effective

[Back to Table of Contents](#)

Upgrade to new version

To upgrade an already installed CHARON-VAX kit to a more recent one:

1. Ensure your license allows you to upgrade to that version. If not, please generate a C2V file and send it to STROMASYS for update. See CHARON-VAX for Linux utilities - 'hasp_srm_view' utility
2. Prepare the new kit RPM files as it is described in "Before Installation" and "Distribution preparation" sections.
3. Stop all running CHARON-VAX instances.
4. Make sure that no template files (i.e. "mv3k6.cfg.template") have been used for your specific configuration otherwise copy those files to a dedicated folder.
5. Login as "root" user.
6. Remove the old CHARON-VAX version as described in the "CHARON-VAX for Linux deinstallation" chapter.
7. Proceed with the same instructions on the new kit installation as described in the "Installation" section.
8. Install the license for the new CHARON-VAX as described in the "License installation" section.
9. Start all the CHARON-VAX services stopped at step #3.

Running CHARON-VAX for Linux

Table of Contents

CHARON-VAX symbolic links

Each model of CHARON-VAX has a symbolic link defined to point to the corresponding CHARON executable (see the table below). If the PATH is correctly defined, you can start a virtual machine by specifying the link followed by the configuration file. This is described further.

Link name	Emulator to run
mv3k196	MicroVAX 3100 Model 96
mv3k198	MicroVAX 3100 Model 98
mv3k6	MicroVAX 3600
mv3k9	MicroVAX 3900
mvii	MicroVAX II
vs4k90	VAXStation 4000 Model 90
vx3k6	VAXserver 3600
vx3k6_128	VAXserver 3600 (128Mb of RAM is available)
vx3k6_512	VAXserver 3600 (512Mb of RAM is available)
vx3k9	VAXserver 3900
vx3k9_128	VAXserver 3900 (128Mb of RAM is available)
vx3k9_512	VAXserver 3900 (512Mb of RAM is available)
vx4k106	VAX 4000 Model 106
vx4k108	VAX 4000 Model 108
vx4k700	VAX 4000 Model 700
vx4k705	VAX 4000 Model 705
vx6k310	VAX 6310
vx6k610	VAX 6610
vx6k620	VAX 6620
vx6k630	VAX 6630
vx6k640	VAX 6640
vx6k650	VAX 6650
vx6k660	VAX 6660

Running CHARON-VAX emulators

It is possible to run one or several instances of CHARON-VAX at the same time if your license allows it.

For multiple instances, please use only absolute paths and unique names to all the files referenced in the configuration file of each CHARON-VAX instance (log, toy clock, nvram files and all the other data such as disk images - all these objects will be explained later; at this stage just pay attention to the files specified in the configuration file) and check the hardware devices are used by only one instance at a time (not shared).

Example:

```
...
set session log="/charon_instances/first/mv3k6.log"
set toy container="/charon_instances/first/mv3k6.dat"

load RQDX3/RQDX3 DUA
set DUA container[0]="/charon_instances/first/mv3k6_boot_disk.vdisk"
...
```

Please refer to the next chapters for more details concerning CHARON-VAX configuration details.

Running from the console

Copy the selected configuration template from the "`/opt/charon/cfg/`" directory to some local file and set the correct privileges for the file to be edited.

Example:

```
$ cp /opt/charon/cfg/mv3k6.cfg.template my_mv3k6.cfg
$ chmod 644 my_mv3k6.cfg
```

Run the virtual machine using this configuration file:

```
$ mv3k6 my_mv3k6.cfg
```

Below is an example of a normal VAX test sequence, followed by the prompt sign (">>>"):

```
KA650-A V5.3, VMB 2.7
Performing normal system tests.

40..39..38..37..36..35..34..33..32..31..30..29..28..27..26..25..
24..23..22..21..20..19..18..17..16..15..14..13..12..11..10..09..
08..07..06..05..04..03..
Tests completed.
>>>
```

The next stage can be either installation of a new VAX/VMS system using a distributive provided by HP or data transfer from some existing VAX system. These possibilities will be discussed in details in the next chapters.

If for some reason CHARON-VAX refuses to start, please look for files with .log extension (CHARON-VAX log files) located in the directory from where CHARON-VAX was started, open them with an editing tool and analyze their content. In most cases those files contain very helpful information on what may possible went wrong.

To exit from the CHARON-VAX emulator use the following methods:

Configuration	How to exit
No change to the template configuration file (not recommended)	Kill the console from where CHARON runs (CTRL-C for example) or kill the CHARON process
Enable "**F6**" button in configuration file to trigger exit from CHARON: ``` #--- # # Uncomment to allow 'F6' to terminate the running emulator. # #--- set OPA0 stop_on = F6 ```	Press the "**F6**" key

⚠ Please note that before stopping CHARON-VAX, a clean shutdown of the operating system running on the virtual machine has to be performed.

Running as system service (daemon)

It is possible to run CHARON-VAX as a daemon (service). In this case the CHARON-VAX process will be detached from its parent process and from the terminal window in which it runs.

Follow the description below to install and execute CHARON-VAX as a daemon:

Installation and start of CHARON-VAX service

1. Copy the sample script "`/opt/charon/bin/charon.service`" ("`/opt/charon/bin/charon`" for Red Hat Linux 6.x) to the "`/usr/lib/systemd/system/`" directory (or to your home directory for Red Hat Enterprise Linux 6.x).

   Example:

Red Hat Linux 6.x	`$ `**`cp`**` /opt/charon/bin/charon /my_services/mv3k6_service` `$ `**`chmod 755`**` /my_services/mv3k6_service`
Red Hat Linux 7.x and CentOS 7.x	`$ `**`cp`**` /opt/charon/bin/charon.service /usr/lib/systemd/system/mv3k6.service` `$ `**`chmod 755`**` /usr/lib/systemd/system/mv3k6.service`

2. Edit the renamed file to replace sample values of the following parameters.

   Example:

Red Hat Linux 6.x	`exec="/opt/charon/bin/mv3k6"`
	`prog="my_mv3k6"`
	`config="/my_services/mv3k6-service.cfg"`

Red Hat Linux 7.x and CentOS 7.x	`ExecStart=/opt/charon/bin/mv3k6 -d /my_services/mv3k6-service.cfg` `WorkingDirectory=/my_services`

   ⚠ "my_mv3k6" is a service name in the example above

3. Create and edit the configuration file ("`/my_services/mv3k6-service.cfg`" in the examples above) the way it was described before and make sure the following pre-requisites are met:
   - OPA0 must be configured as a virtual port or physical console, not as an operator console.

   Example:

   ```
   load virtual_serial_line OPA0 port=10003
   #load operator_console OPA0
   ```

- Use only absolute paths to log, toy clock, nvram files and all the other data such as disk images, etc. The names of the references files must be unique.

    Example:

    ```
    ...
    set session log="/my_services/my_mv3k6.log"
    set toy container="/my_services/my_mv3k6.dat"

    load RQDX3/RQDX3 DUA
    set DUA container[0]="/my_services/mv3k6_daemon_boot_disk.vdisk"
    ...
    ```

- Make sure the same physical devices are not used by other CHARON-VAX daemons and the OPA0 console port number is unique across the CHARON server.

Once configuration file is ready, execute the following commands (based on the examples above) to install and start CHARON-VAX as a daemon:

Red Hat Linux 6.x	# **ln -sf** /my_services/mv3k6_service /etc/init.d/mv3k6_service # **chkconfig** mv3k6_service **on** # **service** mv3k6_service **start**
Red Hat Linux 7.x and CentOS 7.x	# **systemctl enable** mv3k6.service # **systemctl start** mv3k6.service

---

Red Hat Linux 7.x and CentOS 7.x
If you update the /usr/lib/systemd/system/<*my virtual machine*>.service file, the following command must be executed in order to take changes into account:

```
# systemctl daemon-reload
```

Note that a certain delay may appear in finding network licenses by Sentinel Run-time on CHARON-VAX host system startup. If the CHARON-VAX service is starting automatically at host system startup, it may report a "License not found" error and exit.

This problem can be avoided by specifying the "license_key_lookup_retry" parameter in the following way:

```
set session license_key_lookup_retry = "N [, T]"
```

where:

- N = Number of retries looking for the license key (or keys)
- T = Time between retries in seconds. If not specified, 60 seconds is used

Example:

```
set session license_key_lookup_retry = 5
```

In this example, if the license key is not found during initial scan, CHARON-VAX will do 5 more attempts waiting 60 seconds between them.

See General Settings section for more details.

Back to Table of Contents

Stopping CHARON-VAX service

To stop a CHARON-VAX daemon, use the following commands.

Example:

Red Hat Linux 6.x	# **service** mv3k6_service **stop**
Red Hat Linux 7.x and CentOS 7.x	# **systemctl stop** mv3k6

⚠ Please note that before stopping a CHARON-VAX service, a clean shutdown of the operating system running on the virtual machine has to be performed.

Back to Table of Contents

Removing CHARON-VAX service

To remove a CHARON-VAX daemon use the following commands.

Example:

Red Hat Linux 6.x	# **chkconfig** mv3k6_service off # **chkconfig --del** mv3k6_service # **rm -f** /etc/init.d/mv3k6_service
Red Hat Linux 7.x and CentOS 7.x	# **systemctl disable** mv3k6.service # **rm -f** /usr/lib/systemd/system/mv3k6.service

⚠ Please note that before removing a CHARON-VAX service, a clean shutdown of the operating system running on the virtual machine has to be performed and the service has to be stopped.

ℹ Please refer to the next chapters for more details concerning CHARON-VAX configuration details

Back to Table of Contents

CHARON-VAX for Linux configuration

Table of Contents

Creation of your own configuration file using a template

By default, all the CHARON templates are located in the "`/opt/charon/cfg`" folder. Copy the appropriate template configuration file(s) to your home directory or to any directory intended for CHARON-VAX, name them meaningfully and set proper privileges.

Example:

```
$ cp /opt/charon/cfg/mv3k6.cfg.template /my_charon_cfg/my_mv3k6.cfg
$ chmod 644 /my_charon_cfg/my_mv3k6.cfg
```

Please do not edit the original template configuration files since they can be updated or even removed on update/deinstallation of CHARON-VAX

Once the file has been created you can open it in your favorite editing tool and proceed with modifications to reflect the exact features of the system you are going to emulate.

We will review all the parameters step by step issuing some recommendations and guidelines.

Note: the lines preceded by the comment sign "#" inside the configuration files will not be interpreted. You can use this sign to debug your configuration.

VAX model specification

The first configuration statement is the specification of the exact VAX hardware model to emulate.

Example:

```
set session hw_model = MicroVAX_3600
```

You must leave this line untouched.

If you create the CHARON-VAX configuration file from scratch, it must be the very first uncommented line in the configuration file.

Configuration name

The next configuration statement is the "Configuration name" option.

Example:

```
#set session configuration_name = MicroVAX_3600
```

You can optionally uncomment this line to differentiate this CHARON-VAX instance from all others in a multi-instances environment. The configuration name can be any label that is meaningful. It is reported in the log file and is used to set the log file name for rotating log (see further: Rotating log (default)).

Log file parameters

Execution of CHARON-VAX creates one log file or a set of log files reflecting the progress of its start-up and ongoing operation - start and end time of execution, system information, license and configuration details, warnings, reports on problems that may occur, etc. In case of possible problems either with the running CHARON-VAX or the emulated system configuration (such as the absence or malfunction of certain devices), the log file(s) is the primary source to be analyzed for troubleshooting. If it becomes necessary to contact Stromasys for support, the configuration and log files, plus the license number, will be requested to begin problem resolution.

Here is an example of a field test CHARON-VAX log file:

```
20161120:083215:INFO :0:000003A5:hexane.cxx(5312): session is loading built-in configuration "VAX_6610"...
20161120:083215:INFO :0:000003A6:hexane.cxx(5336): session has finished loading built-in configuration
"VAX_6610".
20161120:083215:INFO :0:000003AA:hexane.cxx(5425): session is loading configuration file "vx6k610.cfg"...
20161120:083215:INFO :0:000003AB:hexane.cxx(5455): session has finished loading configuration file
"vx6k610.cfg".
20161120:083215:INFO :0:000003F2:sesmgr.cxx(1394): session: default log file size limit is 4194304 bytes
20161120:083215:INFO :0:0000032B:hexane.cxx(2547): Start request received.
20161120:083216:INFO :0:000003AC:hexane.cxx(1287): session's process affinity is 000000000000000F, system
affinity is 000000000000000F.
20161120:083216:INFO :0:000003D1:hexane.cxx(1541): session's I/O domain affinity is 0000000000000001, CPU
domain affinity is 000000000000000E
20161120:083216:INFO :0:000003BA:ll_sentine( 725): Looking for a license key ...
20161120:083217:INFO :0:000003DC:ll_sentine( 820): ... found license key 1422726238.
20161120:083217:INFO :0:0000024D:hexane.cxx(2688): STROMASYS SA, (C) 2009-2015
20161120:083217:INFO :0:00000350:hexane.cxx(2734): CHARON-VAX (VAX 6000 Model 610), V 4.8 B 18302, Nov 11 2016
/ 000.msc.sanity.tests / 1422726238
20161120:083217:INFO :0:00000336:hexane.cxx(2761): The end user of this software has agreed to STROMASYS' Terms
and Conditions for Software License and Limited Warranty, as described at: http://www.stromasys.com/pub/doc/30-
17-033.pdf
20161120:083217:INFO :0:0000009D:hexane.cxx(2838): License info:
CHARON product code: "CHVAX-460xx-WI".
Licensed to: "MSC".

20161120:083217:INFO :0:00000097:hexane.cxx(2847): OS Environment: Linux 3.19.8-100.fc20.x86_64 #1 SMP Tue May
12 17:08:50 UTC 2015 x86_64.
20161120:083217:INFO :0:00000098:hexane.cxx(2852): Host CPU: GenuineIntel, Family 6, Model 42, Stepping 1,
Intel Xeon E312xx (Sandy Bridge), 1 Cores per Chip, 1 Threads per Core, at ~2593 MHz, 4 cpu's available
20161120:083217:INFO :0:00000099:hexane.cxx(2857): Host Memory: 4096Mb
20161120:083217:INFO :0:00000348:mscpdisk.c( 333): PUA0 is being set OFFLINE
20161120:083217:INFO :0:00000001: tpool.cxx(1370): cpu: The ACE option is omitted; enable ACE as license
default.
20161120:083218:INFO :0:00000133: tpool.cxx(1602): Advanced CPU Emulation (ACE) enabled.
20161120:083218:INFO :0:0000032C:hexane.cxx(2589): "VAX_6610" started.
20161120:083300:INFO :0:00000347:mscpdisk.c(3656): PUA0 is being set ONLINE
container = "/home/charon/Charon/test/performancecomparison-66x0.vdisk"
media_type = ""
geometry = ""
use_io_file_buffering = false
```

The next group of parameters defines the name of the CHARON-VAX log file and how CHARON-VAX will use it:

```
#set session log_method = append
#set session log_method = overwrite
#set session log = "MicroVAX_3600.log"
```

Rotating log (default)

By default CHARON-VAX utilizes a so called "rotating log" mechanism. This means that a new default log file is created each time CHARON starts and can switch to another log file if the size of the log file exceeds 64Kb (this behavior can be changed with the "set session log_file_size" and "set session log_rotation_period" parameters; see more details in the "General Settings" chapter of this guide).

This mode is turned on if all the log parameters above are disabled (commented out) or the "session_log" parameter is pointing to an existing directory rather than to a file. If a directory is specified, the log files will be created in that directory.

The names of the rotating log files are composed as follows:

```
configuration_name-YYYY-MM-DD-hh-mm-ss-xxxxxxxxx.log
```

If the "Configuration name" parameter described before is omitted (commented out), the log name has the following format instead:

```
hw_model-YYYY-MM-DD-hh-mm-ss-xxxxxxxxx.log
```

Note that "xxxxxxxxx" is an increasing decimal number starting from "000000000" to separate log files with the same time of creation.

> Only an existing directory can be used as a value of the "log" parameter.

Single log

Alternatively it is possible to use a single log file: uncomment the "set session log" line and specify the desired CHARON-VAX log file name. Optionally, a path can be added to the log file name. If the path is not specified, the log file is created in the directory from where the guest (emulated machine) is started.

The log file can be extended ("log_method = append") or overwritten ("log_method = overwrite") by CHARON-VAX.

Below is a specification of a CHARON-VAX log file located in the "/my_logs" directory which will be appended each time CHARON-VAX starts:

```
set session log_method = append
set session log = "/my_logs/my_vax.log"
```

TOY, ROM and EEPROM containers

The next objects to be configured are TOY, ROM and EEPROM containers (their presence depends on the VAX model). It is always recommended to enable them. If a container file of the given name does not exist, CHARON-VAX will create it. Specific paths can be added to the file name specification.

TOY means "Time of Year"; its container records time, date and some console parameters while CHARON-VAX is not running. To enable the TOY, uncomment the following line:

```
set toy container="charon.dat"
```

The ROM container stores an intermediate state of the Flash ROM and some console parameters. It is highly recommended to define its location:

```
set rom container="vx4k106.rom"
```

EEPROM stores the NVRAM content. It is highly recommended to define its location:

```
set eeprom container="charon.rom"
```

Emulated memory (RAM) size

The next parameter defines the amount of host memory the chosen CHARON-VAX model reserves for the emulation.

Example:

```
set ram size=64
```

The amount of RAM is specified in MB. It cannot exceed or be lower than certain values specific for each VAX model. It is very important to keep the listed predefined increment between possible memory values.

The following table lists all the parameters per model:

Hardware Model	RAM size (in MB)			
	Min	Max	Default	Increment
MicroVAX_II	1	16	16	1,8,16
MicroVAX_3600	16	64	16	16
MicroVAX_3900	16	64	16	16
VAXserver_3600	16	64	16	16
VAXserver_3900	16	64	16	16
VAXserver_3600_128	32	128	32	32
VAXserver_3900_128	32	128	32	32
MicroVAX_3100_Model_96	16	128	16	16
VAXstation_4000_Model_90	16	128	16	16
VAX_4000_Model_106	16	128	16	16
VAX_6000_Model_310	32	512	32	32
VAXserver_3600_512	32	512	32	32
VAXserver_3900_512	32	512	32	32
MicroVAX_3100_Model_98	16	512	16	16
VAX_4000_Model_108	16	512	16	16
VAX_4000_Model_700	64	512	64	64
VAX_4000_Model_705	64	512	64	64
VAX_6610	128	3584	128	128
VAX_6620	128	3584	128	128
VAX_6630	128	3584	128	128
VAX_6640	128	3584	128	128
VAX_6650	128	3584	128	128
VAX_6660	128	3584	128	128

It is possible to leave the RAM line commented out. In this case the model's default RAM amount is used.

Console

Mapping to system resources

The next step is the specification of the VAX console (OPA0) serial line.

Example:

```
#load physical_serial_line OPA0 line="/dev/ttyN"
#load virtual_serial_line OPA0 port=10003
load operator_console OPA0
```

The goal of this configuration step is to tell CHARON-VAX what host device to use as the virtual system console. The following options are available:

Option	Description
physical_serial_line	Mapping to host serial line, both physical and virtual. Use the following mapping for different types of host serial lines: | Mapping	Description
/dev/tty<N>	Virtual serial lines
/dev/ttyS<N>	Onboard serial lines
/dev/ttyUSB<N>	Modem or usb serial lines adapters | The specific account for running CHARON ("charon") does not allow the usage of physical consoles, "/dev/tty<N>", as CHARON consoles. If you plan to map the CHARON console to "/dev/tty<N>" use only the "root" account to run CHARON.
virtual_serial_line	Mapping to an IP port of CHARON-VAX host. Using this mapping it is possible to connect to the CHARON-VAX console and disconnect from it at any time.
operator_console	Mapping to the current TTY console

The default setting is "operator_console".

Note that the VAX 4000 and MicroVAX 3100 models have a 4-line QUART adapter onboard so their configuration for the console line looks a bit different:

```
#load physical_serial_line/chserial TTA0 line="/dev/tty<N>"
#load virtual_serial_line/chserial TTA0 port=10000
#set quart line[0]=TTA0
...

#load physical_serial_line/chserial TTA2 line="/dev/tty<N>"
#load virtual_serial_line/chserial TTA2 port=10002
#set quart line[2]=TTA2

#load physical_serial_line OPA0 line="/dev/tty<N>"
#load virtual_serial_line OPA0 port=10003

load operator_console OPA0
set quart line[3]=OPA0
```

IF VAX 4000 and MicroVAX 3100 models are used, it is possible to configure up to 4 independent console lines: OPA0, TT0, TT1 and TT2. The main one is OPA0.

⚠ Note there are a number of additional parameters for CHARON-VAX serial line configuration. Follow this link for details.

Exit on pressing F6 key

A hot key can be defined to stop the execution of the CHARON-VAX virtual machine:

```
set OPA0 stop_on = F6
```

It is strongly recommended to uncomment this line to provide CHARON-VAX the ability to exit by pressing the "**F6**" key.

Disk subsystem

The next step is the configuration of the disk subsystem and mapping it to the system resources using the samples given in the template configuration files.

CHARON-VAX supports MSCP, DSSI, CI and SCSI disk controllers. The examples below are for MSCP and SCSI controllers only. DSSI controllers are discussed in details in the following section and CI controllers in this section.

MSCP disk controllers (RQDX3, KDB50, KDM70)

Below is a typical configuration sample for MSCP disk controller RQDX3:

```
load RQDX3 DUA

#set DUA container[0]="<file-name>.vdisk"
#set DUA container[1]="/dev/sdN"
#set DUA container[2]="/dev/srN"
#set DUA container[3]="<file-name>.iso"

#load RQDX3 DUB address=...
#load RQDX3 DUC address=...
```

The first line ("load RQDX3 DUA") loads disk controller RQDX3 with name DUA, followed by 4 lines showing different ways of mapping to the host resources:

Type of mapping	Description
"<file-name>.vdisk"	Mapping to files representing physical disks of the VAX system (disk images). These files can be created from scratch with "mkdskcmd" utility. Data and OS disk backups are transferred from the original system via tapes or network and restored into these container files. Mapping may also include the full path, for example: "/my_disks/my_boot_disk.vdisk"
"/dev/sd<L>"	Mapping to physical disks. "L" is letter here. Be careful not to destroy all the information from the disk dedicated to CHARON-VAX by mistake! These disks can not be formatted by the host OS. ⓘ It is also possible to use not a whole disk, but previously created partitions on it. In this case the syntax is the following: "/dev/sd<L><N>" where N is the number of partition to be used. Since "/dev/sd<L>" addressing is not persistent, so it is strongly recommended to use "/dev/disk/by-id/wwn-*" syntax instead to refer the disk by its WWID - especially in the enviroments utilizing FC and SAN storages (see below).
"/dev/dm-<N>" "/dev/mapper/mpath<N>" "/dev/mapper/disk<N>"	Mapping to multipath disk. ⚠ Be careful not to destroy all the information from the disk dedicated to CHARON-VAX by mistake. These disks must not be formatted by the host OS.
"/dev/disk/by-*"	Mapping to physical disk. • by-id (addressing by the disk ID, for example "/dev/disk/by-id/ata-ST1000DM003-9YN162_S1D01QJ4") • by-label (addressing by the disk label, for example "/dev/disk/by-label/MyStorage") • by-uuid (addressing by the disk UUID, for example "/dev/disk/by-uuid/0e808a2f-cdd3-4944-a245-f729ffd73882") ⚠ Be careful not to destroy all the information from the disk dedicated to CHARON-VAX by mistake. These disks must not be formatted by the host OS.
"/dev/sr<N>"	Mapping to CD-ROMs. There are some variants of this mapping: "/dev/cdrom<N>" or "/dev/cdrom"
"<file-name>.iso"	Mapping to an ISO file for reading distribution CD-ROM images.

The numbers in the square brackets represent unit numbers associated with each container of the MSCP controller. For example, the line of the configuration sample above, referring container #2, creates the "DUA2" disk. The maximum unit number allowed is 9999, significantly more than the original hardware provided.

It is possible to load several RQDX3 controllers DUB, DUC, etc. (see lines 6-7, above) by configuring specific addresses for them on the Qbus. Use the "CONFIGURE" utility available on the VAX console to determine the addresses. Please refer to specific HP documentation for further information.

Please also refer to the HP documentation for information on placement of additional KDM70 controllers on an XMI bus (VAX 6000 models) and additional KDB50 controllers on a BI bus (VAX 6310).

Note that the KDM70 controller is capable of mapping to files representing tapes (tape images) and physical tape devices:

```
set PUA container[600] = "<file-name>.vtape"
set PUA container[601] = "/dev/sg<N>"
```

Follow this link for details of (T)MSCP controllers configuration.

SCSI controller NCR53C94

The VAX 4000 and MicroVAX 3100 have an NCR53C94 SCSI controller onboard for support of different types of SCSI devices including disks and tapes. Optionally a second controller can be added.

Below is a typical configuration template for a preloaded "PKA" NCR53C94 SCSI controller:

```
# Mapping to disk image
load virtual_scsi_disk pka_0 scsi_bus=pka scsi_id=0
set pka_0 container="<file-name>.vdisk"

# Mapping to physical disk
load virtual_scsi_disk pka_1 scsi_bus=pka scsi_id=1
set pka_1 container="/dev/sd<L>"

# Mapping to some SCSI device connected to the host
load physical_scsi_device pka_2 scsi_bus=pka scsi_id=2
set pka_2 container="/dev/sg<N>"

# Mapping to host CD-ROM or DVD-ROM
load virtual_scsi_cdrom pka_3 scsi_bus = pka scsi_id = 3
set pka_3 container = "/dev/cdrom<N>"
#set pka_3 container = "/dev/sr<N>"

# Mapping to *.ISO image
load virtual_scsi_cdrom pka_4 scsi_bus=pka scsi_id=4
set pka_4 container="<file-name>.iso"

# Mapping to tape drive
load physical_scsi_device pka_5 scsi_bus=pka scsi_id=5
set pka_5 container="/dev/sg<N>"

# Mapping to tape image
load virtual_scsi_tape pka_6 scsi_bus=pka scsi_id=6
set pka_6 container="<file-name>.vtape"

# Include this line to get access to "PKB" adapter
include kzdda.cfg

# Mapping to host floppy drive
load virtual_scsi_disk pkb_6 scsi_bus=pkb scsi_id=6
set pkb_6 container="/dev/fd<N>"
```

Note that NCR53C94 SCSI controller mapping to system resources is done via specific auxiliary objects:

Mapping Object	Description
virtual_scsi_disk	Mapping to a file representing VAX disk (disk image) on the host physical disk: • **"<file-name>.vdisk"** These files can be created from scratch with "mkdskcmd" utility. Data and OS disk backups are transferred from the original system via tapes or network and restored into these container files. Mapping may also include the full path, for example: "/my_disks/my_boot_disk.vdisk" • **"/dev/sd<L>"** - name of a physical disk. "L" is letter here. • **"/dev/dm-<N>", "/dev/mapper/mpath<N>", "/dev/mapper/disk<N>"** - for multipath disks. N is 0,1,2... • **"/dev/disk/by-id/..."** - addressing by the disk ID, for example "/dev/disk/by-id/ata-ST1000DM003-9YN162_S1D01QJ4" • **"/dev/disk/by-label/..."** - addressing by the disk label, for example "/dev/disk/by-label/MyStorage" • **"/dev/disk/by-uuid/..."** - addressing by the disk UUID, for example "/dev/disk/by-uuid/0e808a2f-cdd3-4944-a245-f729ffd73882" • **"/dev/df<N>"** - name of host CD-ROM drive. N is 0,1,2... This parameter can be omitted. Be careful not to destroy all the information from the disk dedicated to CHARON-VAX by mistake! These disks can not be formatted by the host OS. > Since "/dev/sd<L>" addressing is not persistent, so it is strongly recommended to use "/dev/disk/by-id/wwn-*" syntax instead to refer the disk by its WWID - especially in the enviroments utilizing FC and SAN storages. ℹ It is also possible to use not a whole disk, but previously created partitions on it. In this case the syntax is the following: "/dev/sd<L><N>" where N is the number of partition to be used.
physical_scsi_device	Mapping to a host SCSI device: • **"/dev/sg<N>"** - name of the SCSI device for direct mapping, for example, a SCSI disk or tape reader.
virtual_scsi_cdrom	Mapping to a host CD-ROM (not only SCSI) or to ISO image: • **"/dev/sr<N>"**, **"/dev/cdrom"**, **"/dev/cdrom<N>"** - name of host CD-ROM drive • **"<file-name>.iso"** - name of ISO image. It may contain the full path, for example: "/my_disks/vms_distributive.iso"
virtual_scsi_tape	Mapping to a file representing tape (tape image). It may contain a path, for example: "/my_tapes/backup.vtape"

Let's look at the syntax of the mapping objects. All of them have several important parameters:

Mapping objects parameters	Description
scsi_bus	The name of the NCR53C94 SCSI controller. A typical value for the first and only preloaded NCR53C94 SCSI controller is "PKA"
scsi_id	SCSI address of this particular mapped device. Note that the preloaded NCR53C94 SCSI controller claims address "7"; addresses 0-6 are vacant and useable. The resulting names of virtual SCSI devices as they are seen in VAX/VMS are made up of the VMS name of the SCSI controller and the device address. For PKA, the device names in VMS will be DKA0, DKA1 etc
container	A keyword for specification of which host device is mapped to a particular virtual SCSI device. It can be disk or tape image, physical disk etc

It is possible to configure another NCR53C94 SCSI controller "PKB" by uncommenting the "include kzdda.cfg" line:

```
include kzdda.cfg

load virtual_scsi_disk pkb_0 scsi_bus=pkb scsi_id=0
set pkb_0 container="<file-name>.vdisk"
...
```

In the example above "pkb_0" virtual SCSI device uses "PKB" controller by specifying a parameter "scsi_bus=pkb"

Note that versions of VAX/VMS older than 5.5-2H4 do not support the optional SCSI controller and might fail to boot if it is loaded.

Follow this link for details of NCR53C94 SCSI controller controllers configuration.

Tape subsystem

Some MSCP and SCSI controllers support tape devices, however CHARON-VAX also emulates specific MSCP tape devices such as TQK50 and TUK50.

Follow this link for more details of (T)MSCP controllers configuration.

TQK50 controller

Example statements to configure TQK50 are shown below:

```
#load TQK50 MUA

#set MUA container[0]="<file-name>.vtape"
#set MUA container[1]="/dev/sg<N>"

#load TQK50 MUB address=...
#load TQK50 MUC address=...
```

The first line ("load TQK50 MUA") loads tape controller TQK50 with a name of MUA. The following 2 lines demonstrate different ways of mapping to host resources:

Type of mapping	Description
"<file-name>.vtape"	Mapping to files representing tapes (tape images). Such files can be created automatically or transferred from physical tapes with "mtd" utility Mapping may also include a full path, for example: "/my_tapes/backup.vtape"
"/dev/sg<N>"	Mapping to host tape devices.

Numbers in the square brackets represent unit numbers associated with each container of the TQK50 controller. For example, line 3 of the configuration sample above creates tape drive "MUA1". The maximum unit number allowed is 9999, significantly more than the original hardware provided

It is possible to load several TQK50 controllers (see the lines 4-5) by configuring specific addresses for them on the Qbus. Use the "CONFIGURE" utility available on the VAX console to determine the addresses. Please refer to specific HP documentation for further information.

TUK50 controller

TUK50 is a UNIBUS controller used by the VAX 6310:

```
load DWBUA UBA vax_bi_node_id = 14

load TUK50 MUA

#set MUA container[0] = "<file-name>.vtape"
#set MUA container[0] = "/dev/sg<N>"
```

The first line loads a UNIBUS BI adapter "DWBUA". Configure then the "TUK50" tape controller the same way as the TQK50.

Serial Lines

CHARON-VAX supports the following serial lines controllers: CXA16, CXB16, CXY08, DHQ11, DHV11, DZV11, DZQ11, DL11, DLV11, DZ11, DHW42-AA, DHW42-BA and DHW42-CA.

All of them are configured according using the following template:

```
#load DHV11/DHV11 TXA
load DHQ11/DHV11 TXA
#load CXY08/DHV11 TXA
#load CXA16/DHV11 TXA
#load CXB16/DHV11 TXA

load physical_serial_line/chserial TXA0 line="/dev/tty0"
#load virtual_serial_line/chserial TXA0 port=10010
set TXA line[0]=TXA0

#load physical_serial_line/chserial TXA1 line="/dev/tty<N>"
#load virtual_serial_line/chserial TXA1 port=10011
#set TXA line[1]=TXA1

...

#load DHV11 TXB address=...
#load DHQ11 TXB address=...
#load CXY08 TXB address=...
#load CXA16 TXB address=...
#load CXB16 TXB address=...
```

The first 5 lines of the example above demonstrate loading serial line controllers of different types. The name of the controller in this example will be "TXA".

Once the controller is loaded it can be mapped to system resources (lines 6-11). The following options are available:

Option	Description
physical_serial_line	Mapping to host serial line, both physical and virtual. Use the following mapping for different types of host serial lines: | Mapping	Description
/dev/tty\<N>	Virtual serial lines
/dev/ttyS\<N>	Onboard serial lines
/dev/ttyUSB\<N>	Modem or usb serial lines adapters	
virtual_serial_line	Mapping to an IP port of CHARON-VAX. This mapping makes it possible to connect to and disconnect from the CHARON-VAX console at any time.

The example above loads a DHQ11 serial line controller with one "TXA0" line mapped to the host virtual serial line "/dev/tty0".

Look at the line "set TXA line[0]=TXA0" in the example: this one and the following lines of similar syntax map the loaded virtual controller ("TXA") to instances of host serial lines ("TXA\<N>").

The number of serial lines possible for each controller depends on its type and corresponds to the HP specification of a given controller.

It is possible to load several CXA16, CXB16, CXY08, DHQ11, DHV11, DZV11, DZQ11, DL11, DLV11 and DZ11 controllers (see the lines 12-16) by configuring specific addresses for them on the Qbus. Use the "CONFIGURE" utility available on the VAX console to determine the addresses. Please refer to specific HP documentation for further information.

VAX 4000 and MicroVAX3100 support DHW42-AA, DHW42-BA and DHW42-CA serial lines adapters:

```
#load DHW42AA/DHV11 TXA
#load DHW42BA/DHV11 TXA
#load DHW42CA/DHV11 TXA

#load physical_serial_line/chserial TXA0 line="/dev/tty<N>"
#load virtual_serial_line/chserial TXA0 port=10010
#set TXA line[0]=TXA0
```

Configuring these adapters is the same as above, except it is possible to load one and only one instance of DHW42-AA, DHW42-BA or DHW42-CA.

Note that additional parameters exist for CHARON-VAX serial lines configuration, follow this link for details.

Back to Table of Contents

Networking

CHARON-VAX supports DEQNA, DESQA, DELQA, DEUNA, DELUA, DEMNA, DEBNI and PMADAA virtual network adapters.

All of them are configured in a similar way:

```
load DELQA/DEQNA XQA

load packet_port/chnetwrk XQA0 interface="eth1"
set XQA interface=XQA0

#load DELQA XQB address=...
#load DELQA XQC address=...
```

In the example above the first line loads a DELQA virtual adapter named "XQA". The following 2 lines map it to the host network interface "eth1".

Note that the mapping is performed in 2 steps:

1. A mapping object "packet_port" named "XQA0" is loaded and connected to the host interface "eth1" so CHARON-VAX will use this interface for its networking
2. The loaded DELQA virtual adapter "XQA" is connected to the "packet_port" object "XQA0"

It is possible to load several DEQNA, DESQA, DELQA, DEUNA and DELUA controllers (see the lines 4-5) by configuring specific addresses for them on the Qbus. Use the "CONFIGURE" utility available on the VAX console to determine the addresses. Please refer to the HP specific documentation for further information.

Some network adapters available in CHARON-VAX are preloaded (for example, the SGEC controller for the MicroVAX 3100 with the predefined name "EZA"), so their configuration is even more simple:

```
load packet_port/chnetwrk EZA0 interface="eth1"
set EZA interface=EZA0
```

> CHARON supports VLAN adapters. If used, proceed with their installation and configuration according to the network adapter vendor User's Guide and then use the resulting VLAN interface the same way as the regular network interface.

Follow this link for more details of CHARON-VAX network controllers configuration.

Back to Table of Contents

Auto boot

CHARON-VAX can be configured to automatically boot an operating system at start up.

The MicroVAX 3100, VAX 6310 and VAX 4000 models boot automatically if the correct boot flags are set at the VAX console level:

```
>>>set halt reboot
```

Please check that the TOY, EEPROM and ROM containers (see above) are enabled so the console changes are saved upon reboots.

The ROM of certain VAX models (MicroVAX II, MicroVAX 3600, MicroVAX 3900, VAXserver 3600 and VAXserver 3900) does not allow the SRM console to accept the above command to enable auto booting. As a workaround, a specific setting can be defined in the configuration file:

```
set bdr boot=auto
```

The CHARON-VAX 6000 models have a similar configuration setting:

```
set xmi boot=auto
```

Host load balance for SMP systems

The VAX 6620 through VAX 6660 models emulate 2-6 CPUs respectively. In this situation, loading of the host system can be tuned with the following configuration file settings:

Setting	Description	Example
affinity	This setting binds the running instance of the emulator CPUs to particular host CPUs. This should be used for soft partitioning host CPU resources or for isolating multiple CHARON instances on the same host from each other. By default the emulator instance allocates as many host CPUs as possible. "Affinity" overrides the default and allows explicit specification of which host CPUs will be used by the instance. Affinity does not reserve the CPU for exclusive use.	`set session affinity = "0, 2, 4, 6"`
n_of_io_cpus	Reserves host CPUs (of those specified by "affinity" parameter, if any) for use by the emulator for I/O handling. By default the emulator instance reserves one third of available host CPUs for I/O processing (round down, at least one). The "n_of_io_cpus" overrides the default by specifying the number of I/O host CPUs explicitly.	`set session n_of_io_cpus = 2`

Migration to CHARON-VAX for Linux

Table of Contents

Introduction

This section describes how to migrate your VAX system to CHARON-VAX. We will use a sample MicroVAX 3600 system to demonstrate the migration procedure. The process is similar for all CHARON-VAX models.

> If the CHARON-VAX based virtual system needs to be created from scratch, refer to the appendix "Configuring devices on the Qbus of a VAX or CHARON-VAX" describing how to find proper Qbus addresses and Vectors for each virtual device.

Collecting information about the source VAX system

The first step is to determine the exact configuration of your VAX hardware in order to create the CHARON-VAX configuration file.

Turn on your source VAX system. At the ">>>" prompt, issue the "show qbus" and "show device" commands:

```
>>>show qbus
Scan of Qbus I/O Space
-200000DC (760334) = FFFF (300) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000DE (760336) = 0B40
-20000124 (760444) = FFFF (304) TQK50/TQK70/TU81E/RV20/KFQSA-TAPE
-20000126 (760446) = 0BC0
-20000140 (760500) = 0080 (310) DHQ11/DHV11/CXA16/CXB16/CXY08
-20000142 (760502) = F081
-20000144 (760504) = DD18
-20000146 (760506) = 0140
-20000148 (760510) = 0000
-2000014A (760512) = 0000
-2000014C (760514) = 8000
-2000014E (760516) = 0000
-20000150 (760520) = 0080 (320) DHQ11/DHV11/CXA16/CXB16/CXY08
-20000152 (760522) = F081
-20000154 (760524) = DD18
-20000156 (760526) = 0140
-20000158 (760530) = 0000
-2000015A (760532) = 0000
-2000015C (760534) = 8000
-2000015E (760536) = 0000
-20001468 (772150) = FFFF (154) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-2000146A (772152) = 0B40
-20001920 (774440) = FF08 (120) DELQA/DEQNA/DESQA
-20001922 (774442) = FF00
-20001924 (774444) = FF01
-20001926 (774446) = FF02
-20001928 (774450) = FFD2
-2000192A (774452) = FF14
-2000192C (774454) = C000
-2000192E (774456) = 1030
-20001940 (774500) = FFFF (260) TQK50/TQK70/TU81E/RV20/KFQSA-TAPE
-20001942 (774502) = 0BC0
-20001F40 (777500) = 0020 (004) IPCR

Scan of Qbus Memory Space
>>>
```

```
>>>show device
UQSSP Disk Controller 0 (772150)
-DUA0 (RZ23)
-DUA1 (RZ24)

UQSSP Disk Controller 1 (760334)
-DUB2 (RZ25)
-DUB3 (RZ26)

UQSSP Tape Controller 0 (774500)
-MUA0 (TK50)

UQSSP Tape Controller 1 (760444)
-MUB3 (TK50)

Ethernet Adapter 0 (774440)
-XQA0 (08-00-01-02-D3-CC)
```

The source VAX configuration in this example is:

Controller	Address	Devices on controller
RQDX3	772150	-DUA0 (RZ23) -DUA1 (RZ24)
RQDX3	760334	-DUB2 (RZ25) -DUB3 (RZ26)
TQK50	774500	-MUA0 (TK50)
TQK50	760444	-MUB3 (TK50)
DHQ11	760520	
DHQ11	760500	
DESQA	774440	-XQA0

To find out the exact types of controllers please refer to documentation on the source VAX system.

Back to Table of Contents

Creation of CHARON-VAX configuration file

Using the above information, the following configuration can be created:

```
...

#
# First RQDX3 controller on address 772150
#
load RQDX3/RQDX3 DUA address=017772150
set DUA container[0]="/my_disks/rz23.vdisk"
set DUA container[1]="/my_disks/rz24.vdisk"

#
# Second RQDX3 controller on address 760334
#
load RQDX3/RQDX3 DUB address=017760334
set DUB container[2]="/my_disks/rz25.vdisk"
set DUB container[3]="/my_disks/rz26.vdisk"

#
# First TQK50 controller on address 774500
#
load TQK50/TQK50 MUA address=017774500
set MUA container[0]="/my_tapes/tape1.vtape"

#
# Second TQK50 controller on address 760444
#
load TQK50/TQK50 MUB address=017760444
set MUB container[3]="/my_tapes/tape2.vtape"

#
# First DHQ11 controller on address 760500
#
load DHQ11/DHV11 TXA address=017760500
load virtual_serial_line/chserial TXA0 port=10010
set TXA line[0]=TXA0

#
# Second DHQ11 controller on address 760520
#
load DHQ11/DHV11 TXB address=017760520
load virtual_serial_line/chserial TXB0 port=10011
set TXB line[0]=TXB0

#
# DESQA controller on address 774440
#
load DESQA/DEQNA XQA address=017774440 interface=XQA0
load packet_port/chnetwrk XQA0 interface="eth1"

...
```

Note the Qbus addresses specification: the number is prefixed with "0", meaning it is an octal value. The number of digits reflects the 22 bits Qbus architecture.

Additional DHQ11 serial lines can be mapped later. For now, only 2 lines are configured, they are mapped to IP ports 10010 and 10011.

DESQA is mapped to the "eth1" network interface. This interface will be used for CHARON-VAX (see the Installation section for more details) on this particular host.

Making disk images

In our example, possible mappings of the RQDX3 and TQK50 tapes include physical devices and disk and tape images. Tape images have not to be manually created whereas you have to provision disk images, as described below.

Our example creates disk images of the original physical type. In reality, this step is the best opportunity in the migration to provision larger disks to get extra storage space.

Create special directories for storing disk and tape images, as needed. These directories are referenced in the sample configuration file above.

```
$ mkdir /my_disks
$ mkdir /my_tapes
```

Next, create the disk images using the "mkdskcmd" utility:

```
$ mkdskcmd -d rz24 -o /my_disks/rz24.vdisk
Please wait...
100% done
Success.
$ mkdskcmd -d rz25 -o /my_disks/rz25.vdisk
Please wait...
100% done
Success.
$ mkdskcmd -d rz26 -o /my_disks/rz26.vdisk
Please wait...
100% done
Success.
$ mkdskcmd -d rz27 -o /my_disks/rz27.vdisk
Please wait...
100% done
Success.
```

Installation of VAX operating system

The next step is to transfer the data from the source VAX system to CHARON-VAX. The easiest way to do it is via backups over the network. For this operation we need a bootable, network-enabled operating system on a CHARON-VAX disk image or physical disk.

The example configures the CHARON-VAX MicroVAX 3600 system for installation of VAX/VMS from a distribution CD-ROM (usually it is "/dev/cdrom" if the host has only one CD-ROM drive):

```
#
# First RQDX3 controller on address 772150 with addition of 3 units: a disk for VAX/VMS, storage disk and
CD-ROM drive
#
load RQDX3/RQDX3 DUA address=017772150
set DUA container[0]="/my_disks/rz23.vdisk"
set DUA container[1]="/my_disks/rz24.vdisk"
set DUA container[2]="/my_disks/new_vms_system.vdisk"
set DUA container[3]="/my_disks/backup_storage.vdisk"
set DUA container[4]="/dev/cdrom"
```

⚠ DUA3 will be the disk where all the source disks will be copied so its size needs to be large enough to store all the disk backup images.

Create an empty disk image for installation of VAX/VMS and another one for storing backups from the source VAX system as it is shown in the section above:

```
$ mkdskcmd -d rz27 -o /my_disks/new_vms_system.vdisk
Please wait...
100% done
Success.
$ mkdskcmd -d rz59 -o /my_disks/backup_storage.vdisk
Please wait...
100% done
Success.
```

Run CHARON-VAX and boot from "dua4" ("migration.cfg" is the configuration file we use in this example):

```
$ mv3k6 migration.cfg
KA650-A V5.3, VMB 2.7
Performing normal system tests.
40..39..38..37..36..35..34..33..32..31..30..29..28..27..26..25..
24..23..22..21..20..19..18..17..16..15..14..13..12..11..10..09..
08..07..06..05..04..03..
Tests completed.
>>>boot dua4
```

Install VAX/VMS including DECnet on "dua2". The DECnet address must belong to the same area as the source VAX system.

Login to the newly installed VAX/VMS system and Initialize the disk intended for backups storage. Let's assume it's prompt is "newvms$ ".

```
newvms$ INIT DUA300: SCRATCH
newvms$ MOUNT/SYS/NOASSIST DUA300: SCRATCH
```

Making remote backups

Now we are ready to create disk backups from the source VAX system to CHARON-VAX.

Boot the CHARON-VAX virtual machine and make sure that the source VAX system is reachable via DECnet.

Login to the source VAX system, shut down all the batch queues, kick off the users, stop all applications and close databases if any. The commands listed in the SYS$MANAGER:SYSHUTDWN.COM file may be helpful. The goal is to close as many files as possible. The system disk will have several files opened (pagefile, swapfile, etc.), this is a normal situation.

ℹ The use of the "SHOW DEVICE /FILES" command would be of help to know files opened on a disk

Let's assume the CHARON-AXP system is node 1.400 in this example. Issue then the following commands from the source VAX system whose prompt is set to "source$ ":

```
source$ BACKUP/IMAGE/IGNORE=INTER DUA0: 1.400"username password"::DUA3:[000000]DUA0.BCK/SAVE
source$ BACKUP/IMAGE/IGNORE=INTER DUA1: 1.400"username password"::DUA3:[000000]DUA1.BCK/SAVE
source$ BACKUP/IMAGE/IGNORE=INTER DUB0: 1.400"username password"::DUA3:[000000]DUB0.BCK/SAVE
source$ BACKUP/IMAGE/IGNORE=INTER DUB1: 1.400"username password"::DUA3:[000000]DUB1.BCK/SAVE
```

When the backup procedure will be completed, the disk "DUA3" of the CHARON-VAX virtual machine will contain 4 savesets: "DUA0.BCK", "DUA1.BCK", "DUB0.BCK" and "DUB1.BCK"

Restore backups to CHARON-VAX disks

Next, restore the new savesets to their corresponding virtual disks. Login to CHARON-VAX and issue this sequence of commands to restore all the savesets created in the previous step:

```
newvms$ MOUNT/FOR DUA0:
newvms$ BACKUP/IMAGE DUA3:[000000]DUA0.BCK/SAVE DUA0:
newvms$ DISMOUNT DUA0:

newvms$ MOUNT/FOR DUA1:
newvms$ BACKUP/IMAGE DUA3:[000000]DUA1.BCK/SAVE DUA1:
newvms$ DISMOUNT DUA1:

newvms$ MOUNT/FOR DUB0:
newvms$ BACKUP/IMAGE DUA3:[000000]DUB0.BCK/SAVE DUB0:
newvms$ DISMOUNT DUB0:

newvms$ MOUNT/FOR DUB1:
newvms$ BACKUP/IMAGE DUA3:[000000]DUB1.BCK/SAVE DUB1:
newvms$ DISMOUNT DUB1:
```

If you are going to have the CHARON-VAX and the original physical VAX system on the network at the same time, you must change the network identity of one (usually the CHARON-VAX).

The easiest way is to boot the CHARON-VAX virtual ized system on the restored system disk with the network disabled and to configure new addresses, as needed.

ⓘ The NIC can be disabled with a "disabled" statement in the CHARON configuration file.

Then Enable the network and reboot.

Back to Table of Contents

Alternative ways of data transfer

Some alternative methods of data transfer are also possible. For example:

- Connect a SCSI tape drive to the CHARON-VAX host via a PCI card
    - Map the tape drive in the CHARON-VAX configuration file
        - a. Restore the source VAX system backups from tape to disk images via VAX/VMS running on CHARON-VAX.
        - b. Boot from standalone backups and restore the content to CHARON-VAX virtual disks.
    - Dump the source VAX system backups to tape images with the "mtd" utility and:
        - a. Boot from the freshly installed VAX/VMS system and restore the tape images to CHARON-VAX virtual disks.
        - b. Boot from standalone backups and restore the content to CHARON-VAX virtual disks.
- Create a network cluster between the source VAX system and CHARON-VAX (it is possible to use the source system as a boot server) then perform backups from one disk to another:

```
$ BACKUP/IMAGE/IGNORE=INTER REAL$DUA0: DUA0:
```

Back to Table of Contents

CHARON-VAX for Linux DSSI cluster

Table of Contents

Introduction

This section will describe how to configure DSSI cluster in CHARON-VAX for Linux.

General description

The DSSI storage subsystem for the CHARON VAX 4000 106, 108, 700 and 705 models is based on the emulation of "SHAC" host adapters. Routing of SCS cluster information among the emulated "SHAC" host adapters of multiple nodes is done via separate TCP/IP links.

The DSSI storage subsystem is functionally emulated and operates at a much higher throughput than the original hardware. Connection to physical DSSI hardware is neither possible nor planned for future releases.

The current version of DSSI emulation for CHARON-VAX supports up to 3 VAX nodes in a virtual DSSI cluster and handles a maximum cluster size of 8 nodes. A single virtual DSSI network supports up to 256 storage elements.

For more details on DSSI configuration follow this link.

Configuration steps

To create a CHARON-VAX DSSI cluster, the following elements must be configured:
1. "SHAC" host adapter
2. "HSD50" storage controller

DSSI hardware topology is emulated by establishing TCP/IP channels between the emulated SHAC host adapters of each CHARON-VAX system. The emulated HSD50 storage controllers are then connected to every SHAC host adapter in the virtual DSSI network.

Cluster operation requires (virtual) disks that are simultaneously accessible by all CHARON-VAX nodes involved. This can be implemented for instance by using a properly configured iSCSI initiator / target structure or a fiber channel storage back-end. Disks on a multiport SCSI switch are not acceptable, as a SCSI switch does not provide true simultaneous access to multiple nodes.

Steps to configure DSSI cluster:

1. Set unique ID for SHAC controller of each node

Configuration	IDs of SHAC is set using console (and stored in ROM file, so make sure the ROM file is specified in configuration file) in the following way:
	``` >>>sho dssi_id DSSI_ID Bus 0/A = 1 DSSI_ID Bus 1/B = 1 >>>set dssi_id 0 2 >>>set dssi_id 1 2 ```
Description	⚠ Note that value of "dssi_id" must be unique for each SHAC instances and HSD50 controllers must not use them; for example if 0 and 1 is used as SHAC ID of different VAX nodes the HDS50 controller instance cannot use ID 0 and 1.

2. Configure preloaded SHAC adapters PAA

Configuration	``` set PAA port[<ID of the node to connect to>]=<port to receive connection from the node specified by ID> host[<ID of the node to connect to>]="<connected node host>:<connected node port>" ```
	Example:
	``` set PAA port[2]=11012 host[2]="pollux:11021" ```
	In this example a VAX node SHAC adapter connects to a node "pollux" (having ID=2) to its port 11021 and expects connection from it on the port 11012.
Description	The second step is to interconnect SHAC adapters of different nodes.
	Use ID of the nodes as a reference for interconnections as it is shown in the example at left.
	Detailed meaning of the parameters is explained in "Example 1" section below

3. Load HSD50 adapter

Configuration	``` load HSD50 <instance name> dssi_host=<SHAC instance> dssi_node_id=<HSD50 node ID> ```
	Example:
	``` load HSD50 DISKS dssi_host=PAA dssi_node_id=3 ```
	In this example HSD50 controller instance "DISKS" is loaded, connected to "PAA" SHAC controller and assigned with ID 3.
Description	Load HSD50 instance, assign ID for it and connect it to "PAA" SHAC adapter.
	For each member of cluster an identical definition of HSD50 controllers must be loaded. It is convenient to introduce a specific include configuration file containing the same configuration commands as it is shown in "Example 2" section below.
	⚠ Note that <HSD50 node ID> parameter must be unique for each SHAC and HSD50 instances; for example if 0 and 1 are used as IDs for nodes (see the Step 1) the HDS50 instance cannot use 0 or 1, but it can be set to 3, for example.

4. Set SCS system ID and allocation class

Configuration	`set <HSD50 instance name> scs_system_id=<SCS ID> mscp_allocation_class=<allocation class>`
	Example: `set DISKS scs_system_id=3238746238 mscp_allocation_class=1` In this example HSD50 instance "DISKS" (configured at the previous step) is assigned with SCS ID "3238746238" and MSCP allocation class 1.
Description	Set SCS system ID and the MSCP allocation class. ⚠️ These setting must be the same for all HSD50 controllers loaded! Note that the MSCP allocation class must be different from the allocation class of the VAX node (excuding some special cases when they can be the same - please refer to OpenVMS Guides on Clusters configuration for more details). For nodes it is set using MODPARAMS.DAT and AUTOGEN Refer to OpenVMS User's Guides for details

5. Configure mapping to the system resources

Configuration	`set <HSD50 instance name> container[<unit number>] = <mapping>`
	Example: `set DISKS container[0]="/mnt/share/dua0-rz24-vms-v6.2.vdisk"` In this example the Unit 0 of the HSD50 controller instance "DISKS" is mapped to some disk image, located in some shared directory on disk server.
Description	The final step is specifying HSD50 mapping to the system resources. Since the disks should be distributed between different VAX nodes they should be located as disk images on some shared space or - alternatively - CHARON host should use such mechanism as iSCSI to address the common disks on some disk storage. See the sections below for different examples of the mapping

It is advisable to start any field test with implementing the cluster examples provided below

Example 1: Dual node DSSI cluster with 4 shared disks

To setup two emulated VAX 4000 Model 108 nodes, we need two host machines, preferably running the same version of Linux.

Assume that these host systems have network host names *CASTOR* and *POLLUX* in the host TCP/IP network.

The following are CHARON-VAX configuration files for the emulated VAX 4000 Model 108 nodes running on *CASTOR* and *POLLUX*:

CASTOR **node**

```
...
set PAA port2=11012 host[2]="pollux:11021"

load HSD50 DISKS dssi_host=PAA dssi_node_id=3

set DISKS scs_system_id=3238746238 mscp_allocation_class=1

set DISKS container[0]="/mnt/share/dua0-rz24-vms-v6.2.vdisk"
set DISKS container[1]="/mnt/share/dua1-rz24-vms-v6.2.vdisk"
set DISKS container[2]="/mnt/share/dua2-rz24-vms-v6.2.vdisk"
set DISKS container[3]="/mnt/share/dua3-rz24-vms-v6.2.vdisk"
...
```

POLLUX **node**

```
...
set PAA port1=11021 host[1]="castor:11012"

load HSD50 DISKS dssi_host=PAA dssi_node_id=3

set DISKS scs_system_id=3238746238 mscp_allocation_class=1

set DISKS container[0]="/mnt/share/dua0-rz24-vms-v6.2.vdisk"
set DISKS container[1]="/mnt/share/dua1-rz24-vms-v6.2.vdisk"
set DISKS container[2]="/mnt/share/dua2-rz24-vms-v6.2.vdisk"
set DISKS container[3]="/mnt/share/dua3-rz24-vms-v6.2.vdisk"

...
```

Let's review both configurations step-by-step.

1. The first line of both configuration files establishes parameters for the preloaded "PAA" SHAC host adapter. Only 2 parameters of SHAC are important for us in this situation:

Parameter	Description
port	An integer value that specifies the TCP/IP port number on which an emulated SHAC host adapter listens for connections from another emulated SHAC host adapter. Possible port values are from 1024 through 32767.
host	A string value that specifies the TCP/IP host name (and optionally TCP/IP port number) to connect to another emulated SHAC host adapter. The syntax for the string is "host-name[:port-no]", with possible values for "port-no" in the range from 1024 through 32767.

Thus, *CASTOR* connects to *POLLUX*'s port 11021 and listens for *POLLUX*'s connection on port 11012, *POLLUX* connects to *CASTOR*'s port 11012 and listens for *CASTOR*'s connection on port 11021

2. Second and third lines of both configuration files are for loading "DISKS" HSD50 storage controller and its parametrization:

Parameter	Description
dssi_host	A string value that specifies an instance name of the emulated SHAC host adapter serving virtual DSSI network.
	If this value is not set, CHARON-VAX tries to locate the host adapter automatically. This automatic lookup works only if the CHARON-VAX configuration has exactly one instance of emulated SHAC host adapter.
dssi_node_id	An integer value that specifies address of emulated HSD50 storage controller on virtual DSSI network. Possible values are from 0 through 7 (initially set to 0).
scs_system_id	A string value that specifies *SCSNODENAME* of emulated HSD50 storage controller.
	The string is up to 10 characters long. Possible characters are uppercase letters A through Z , figures 0 through 9.
mscp_allocation_class	An integer value that specifies *ALLOCLASS* of emulated HSD50 storage controller.
	Possible values are from 0 through 255 (initially set to 0).

> In both configuration files, the data related to the emulated HSD50 storage controller "DISKS" **must be identical**. Not following this rule can cause data corruption on the (virtual) disks.

3. The next lines demonstrate mapping "DISKS" HSD50 storage controller to disk images, shared between both hosts.. A "container" parameter is used for this purpose. This example assumes that all disk images are accessible from both host machines via network share (NFS, SAMBA) or some other realization.

Example 2: Triple node DSSI cluster with multiple iSCSI disks

In this example we assume that all three host systems have a iSCSI initiator and are connected to a common iSCSI server. The iSCSI disk server provides 8 virtual disks with R/W access on all hosts. These disks are configured as "/dev/sdc0" ... "/dev/sdc7" on each of the host machines.

Since the storage configuration must be identical on all three nodes, it is recommended to describe the storage structure in a separate configuration file (to be included in each CHARON-VAX configuration file with "include" instruction and name of the configuration file "disksets.cfg") and store it on a common network share ("/mnt/share") (NFS, SAMBA, etc):

```
load HSD50 DISKS1 dssi_host=PAA dssi_node_id=4

set DISKS1 scs_system_id=3238746238 mscp_allocation_class=1

set DISKS1 container[1]="/dev/sdc0"
set DISKS1 container[2]="/dev/sdc1"
set DISKS1 container[3]="/dev/sdc2"
set DISKS1 container[4]="/dev/sdc3"

load HSD50 DISKS2 dssi_host=PAA dssi_node_id=5

set DISKS2 scs_system_id=1256412654 mscp_allocation_class=2

set DISKS2 container[5]="/dev/sdc4"
set DISKS2 container[6]="/dev/sdc5"
set DISKS2 container[7]="/dev/sdc6"
set DISKS2 container[8]="/dev/sdc7"
```

CHARON-VAX configuration file for the emulated VAX 4000 Model 108 node running on *HOST001* is as follows:

```
...

set PAA port[2]=11012 host[2]="host002:11021"
set PAA port[3]=11013 host[3]="host003:11031

include /mnt/share/disksets.cfg
...
```

CHARON-VAX configuration file for the emulated VAX 4000 Model 108 node running on *HOST002* is as follows:

```
...
set PAA port[1]=11021 host[1]="host001:11012"
set PAA port[3]=11023 host[3]="host003:11032"

include /mnt/share/disksets.cfg
...
```

CHARON-VAX configuration file for the emulated VAX 4000 Model 108 node running on *HOST003* is as follows:

```
...
set PAA port[1]=11031 host[1]="host001:11013"
set PAA port[2]=11032 host[2]="host002:11023"

include /mnt/share/disksets.cfg
...
```

Back to Table of Contents

CHARON-VAX for Linux CI cluster

Table of Contents

Introduction

This section describes how to configure a CHARON-VAX for Linux CI cluster.

General description

The virtual CIXCD is the functional equivalent of a hardware CIXCD host adapter, with the exception that there is no physical layer to connect to a hardware CI infrastructure. Since the current host hardware is much faster than the physical CI implementation, such a connection - if it were possible - would greatly limit the virtual system throughput.

For data storage, the CIXCD connects to one or more virtual HSJ50 controllers that are loaded as a separate components in the configuration file. To configure VAX CI clusters, the virtual CIXCDs of the multiple CHARON-VAX/66X0 instances are interconnected via TCP/IP links.

It is advisable to start any field test based on the cluster examples provided below

Configuring (large) virtual VAX CI clusters requires many configurable parameters and a replicated identical definition of the shared virtual HSJ50 storage controllers in each virtual VAX instance.

The current CI implementation for CHARON-VAX/66x0 supports up to 8 VAX nodes in a virtual CI cluster and handles a maximum cluster size of 128 nodes. A single virtual *CI* network supports up to 256 storage elements.

For more details on CI configuration follow this link.

Configuration steps

To create CHARON-VAX CI cluster, both of the two elements must be configured:
1. "CIXCD" host adapter
2. "HSJ50" storage controller

CI hardware topology is emulated by establishing TCP/IP channels between the emulated CIXCD host adapters of each CHARON-VAX system. The emulated HSJ50 storage controllers are then connected to every CIXCD host adapter in the virtual CI network.

Cluster operation requires (virtual) disks that are simultaneously accessible by all CHARON-VAX nodes involved. This can be implemented for instance by using a properly configured iSCSI initiator / target structure or a fiber channel storage back-end. Disks on a multiport SCSI switch are not acceptable, as a SCSI switch does not provide true simultaneous access to multiple nodes.

> It is advisable to start any field test with implementing the cluster examples provided below

Example 1: Dual node CI cluster with 4 shared disks

To setup two emulated VAX 6610 nodes, we need two host machines, preferably running the same version of Linux.

Assume that these host systems have network host names *CASTOR* and *POLLUX* in the host TCP/IP network.

The following are CHARON-VAX configuration files for the emulated VAX 6610 nodes running on *CASTOR* and *POLLUX*:

CASTOR node

```
...
load CIXCD PAA ci_node_id=1

set PAA port[2]=11012 host[2]="pollux:11021"

load HSJ50 DISKS ci_host=PAA ci_node_id=101

set DISKS scs_system_id=3238746238 mscp_allocation_class=1

set DISKS container[0]="/mnt/share/dua0-rz24-vms-v6.2.vdisk"
set DISKS container[1]="/mnt/share/dua1-rz24-vms-v6.2.vdisk"
set DISKS container[2]="/mnt/share/dua2-rz24-vms-v6.2.vdisk"
set DISKS container[3]="/mnt/share/dua3-rz24-vms-v6.2.vdisk"
...
```

POLLUX node

```
...
load CIXCD PAA ci_node_id=2

set PAA port[1]=11021 host[1]="castor:11012"

load HSJ50 DISKS ci_host=PAA ci_node_id=101

set DISKS scs_system_id=3238746238 mscp_allocation_class=1

set DISKS container[0]="/mnt/share/dua0-rz24-vms-v6.2.vdisk"
set DISKS container[1]="/mnt/share/dua1-rz24-vms-v6.2.vdisk"
set DISKS container[2]="/mnt/share/dua2-rz24-vms-v6.2.vdisk"
set DISKS container[3]="/mnt/share/dua3-rz24-vms-v6.2.vdisk"
...
```

Let's review both configurations step-by-step.

1. The first two lines of both configuration files load and establish parameters for the "PAA" CIXCD host adapter. Only 3 CIXCD parameters are important for us in this situation:

Parameter	Description
ci_node_id	An integer value that specifies the address of the virtual CIXCD host adapter on the virtual CI network. Possible values are from 0 through 127 (Initially set to 127).
port	An integer value that specifies the TCP/IP port number at which the emulated CIXCD host adapter listens for connections from another emulated CIXCD host adapter with a certain CI node id. Possible values are from 1024 through 32767.
host	A string value that specifies the TCP/IP host name (and optional TCP/IP port number) to connect to another emulated CIXCD host adapter with certain CI node. The syntax for the string is "host-name[:port-no]", with possible values for "port-no" in the range from 1024 through 32767.

Thus, *CASTOR* connects to *POLLUX*'s port 11021 and listens for *POLLUX*'s connection on port 11012, *POLLUX* connects to *CASTOR*'s port 11012 and listens for *CASTOR*'s connection on port 11021

2. The third and fourth lines of both configuration file "DISKS" HSJ50 storage controller and its parameters:

Parameter	Description
ci_host	A string value that specifies an instance name of the emulated CIXCD host adapter serving the virtual CI network.
	If this value is not set, CHARON-VAX tries to locate the host adapter automatically. This automatic lookup works only if the CHARON-VAX configuration has exactly one instance of an emulated CIXCD host adapter.
ci_node_id	An integer value that specifies the address of the emulated HSJ50 storage controller on a virtual CI network. Possible values are from 0 through 7 (initially set to 0).
scs_system_id	A string value that specifies *the SCSNODENAME* of the emulated HSJ50 storage controller.
	The string is up to 10 characters long. Possible characters are uppercase letters A through Z , figures 0 through 9.
mscp_allocation_class	An integer value that specifies the *ALLOCLASS* of an emulated HSJ50 storage controller.
	Possible values are from 0 through 255 (initially set to 0).

> In both configuration files, the data related to the emulated HSJ50 storage controller "DISKS" **must be identical**. Not following this rule can cause data corruption on the (virtual) disks.

3. The next lines demonstrate mapping the "DISKS" HSJ50 storage controller to disk images, shared between both hosts. A "container" parameter is used for this purpose. This example assumes that all disk images are accessible from both host machines via network share (NFS, SAMBA) or some other realization.

Back to Table of Contents

Example 2: Triple node CI cluster with multiple iSCSI disks

In this example we assume that all three host systems have a iSCSI initiator and are connected to a common iSCSI server. The iSCSI disk server provides 8 virtual disks with R/W access on all hosts. These disks are configured as "/dev/sdc0" ... "/dev/sdc7" on each of the host machines.

Since the storage configuration must be identical on all three nodes, it is recommended to describe the storage structure in a separate configuration file (to be included in each CHARON-VAX configuration file with "include" instruction and name of the configuration file "disksets.cfg") and store it on a common network share ("/mnt/share") (NFS, SAMBA, etc):

```
load HSJ50 DISKS1 ci_node_id=4

set DISKS1 scs_system_id=3238746238 mscp_allocation_class=1

set DISKS1 container[1]="/dev/sdc0"
set DISKS1 container[2]="/dev/sdc1"
set DISKS1 container[3]="/dev/sdc2"
set DISKS1 container[4]="/dev/sdc3"

load HSJ50 DISKS2 ci_node_id=5

set DISKS2 scs_system_id=1256412654 mscp_allocation_class=2

set DISKS2 container[5]="/dev/sdc4"
set DISKS2 container[6]="/dev/sdc5"
set DISKS2 container[7]="/dev/sdc6"
set DISKS2 container[8]="/dev/sdc7"
```

CHARON-VAX configuration file for the emulated VAX 6610 node running on *HOST001* is as follows:

```
...
load CIXCD PAA ci_node_id=1

set PAA port[2]=11012 host[2]="host002:11021"
set PAA port[3]=11013 host[3]="host003:11031

include /mnt/share/disksets.cfg
...
```

CHARON-VAX configuration file for the emulated VAX 6610 node running on *HOST002* is as follows:

```
...
load CIXCD PAA ci_node_id=2

set PAA port[1]=11021 host[1]="host001:11012"
set PAA port[3]=11023 host[3]="host003:11032"

include /mnt/share/disksets.cfg
...
```

CHARON-VAX configuration file for the emulated VAX 6610 node running on *HOST003* is as follows:

```
...
load CIXCD PAA ci_node_id=3

set PAA port[1]=11031 host[1]="host001:11013"
set PAA port[2]=11032 host[2]="host002:11023"

include /mnt/share/disksets.cfg
...
```

Back to Table of Contents

CHARON-VAX for Linux virtual network

Table of Contents

General description

It is strongly recommended to use only physical network adapters for CHARON-VAX networking to gain maximum performances. In situations where the host has only one network adapter, you can use Linux virtual network Interfaces ("TUN/TAP") and map individual CHARON-VAX instances to their own virtual interfaces.

There are 2 ways to create the Linux virtual network Interfaces ("TUN/TAP"):

- Using the "ncu" utility
- Manually

Using "ncu" utility to establish CHARON virtual network

Login a root and start the "ncu" utility:

```
# ncu

CHARON Network Configuration Utility, STROMASYS (c) 2016 Version 1.6


Interfaces Dedicated to State
---------- ------------ -----
eth0 host connected to host
eth1 host disconnected from host
lo host unmanaged from host
================================================================
bridge name bridge id STP enabled interfaces
========================= VLAN ================================
================================================================
select action:
1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit

:> 3
```

Enter "3" to create a bridge between the host physical network adapter and the Linux virtual network Interfaces (TAP) and specify the physical network interface ("eth1" in our example) and the number of the virtual network Interfaces to be created (2 in our example):

```
Specify the interface to be used for BRIDGE:eth1
How many tap should be created:2
Forming the bridge: ..1..2..3..4..5.. addif tap0 .. addif tap1 ..7..8 done!
Formed bridge br0_eth1 attached over eth1...

select action:
1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit

:> 7
```

Now enter "7" to see the created virtual interfaces:

```
Interfaces Dedicated to State
---------- ------------ -----
eth0 host connected to host
eth1 bridge disconnected from bridge
lo host unmanaged from host
tap0 CHARON connected to host
tap1 bridge connected to bridge
================================================================
bridge name     bridge id            STP enabled    interfaces
br0_eth1        8000.22314588acac    no             eth1
tap0
tap1
========================= VLAN =================================

================================================================

select action:
1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit

:> 8
```

In the example above we see 2 virtual network Interfaces, "tap0" and "tap1", connected to the created bridge. The physical network interface "eth1" is used for the bridge to the virtual network interfaces.

The interfaces "tap0" and "tap1" are ready to be used in CHARON configurations, they do not need to be additionally dedicated to CHARON.

Enter "8" to quit the "ncu" utility.

Manual configuration of CHARON virtual network

Host preparation

1. Login as "root" user.

2. Configure the physical network interface to run in promiscuous mode using the following command. This interface will be dedicated to the whole network bridge (created later).

```
# ifconfig eth<N> 0.0.0.0 promisc up
```

The promiscuous mode allows the physical (or virtual) network interface to accept the entire volume of incoming packets. This mode is essential for consistency of the information transfer.

3. In case the firewall is enabled on the host system, the following command should be executed to allow the bridge to forward IP packets:

```
# /sbin/iptable -I FORWARD -m physdev --physdev-is-bridged -j ACCEPT
```

This command can also be performed from the bridge configuration script. It has to be executed each time the *iptables* service is (re)started.

It is also possible to make this setting system-wide. Either:
    a. Issue the given command from the firewall control panel.
    b. Add the following line to the end of the "`/etc/sysconfig/iptables`" file:

```
-I FORWARD -m physdev --physdev-is-bridged -j ACCEPT
```

Virtual interface creation

The creation of the desired number of virtual network interfaces (TAPs) can be performed in the following way:

```
# tunctl [-t tap<N>]
```

where "tap<N>" is a name of an instance of the virtual network interface, i.e. "tap0", "tap1" etc.

Once each virtual network interface instance is created it must be set to promiscuous mode:

```
# /sbin/ifconfig tap<N> promisc up
```

Bridge creation

To interconnect the physical and virtual network interfaces created in the previous step, the network bridge must be introduced in the following way:

```
# /usr/sbin/brctl addbr br0
```

where "*br0*" stands for a name of the created bridge.

Now it is possible to add the network interfaces to the created bridge:

```
# /usr/sbin/brctl addif br0 eth<N>
# /usr/sbin/brctl addif br0 tap0
…
# /usr/sbin/brctl addif br0 tap<N>
```

Example:

```
# /usr/sbin/brctl addif br0 eth1

# /usr/sbin/brctl addif br0 tap0
```

The proposed configuration assumes one and only one network bridge so loops are not possible. It is required to turn off the spanning tree protocol with the following command:

```
# /usr/sbin/brctl stp br0 off
```

Starting bridge

To start the created bridge "*br0*", use the following command:

```
# /sbin/ifconfig br0 up
```

Usage of the virtual interface in CHARON-VAX configuration

Once the "*tap<N>*" interfaces have been created, the load command maps those interfaces to CHARON-VAX:

```
...
load tap_port/chnetwrk XQA0 interface="tap<N>"
...
```

CHARON-VAX for Linux licensing

Table of Contents

General description

The CHARON-VAX product is protected by licenses, issued by STROMASYS for each customer individually. The CHARON-VAX license defines all the specifics of the particular CHARON-VAX distribution and its usage.

The license is implemented in the form of a hardware dongle (a Sentinel HASP key) or a software bound to the hardware. Please be careful with your license key. In case of loss or damage, CHARON-VAX will not run or start unless the license key is replaced. For extra protection, STROMASYS recommends the use of a backup license key (purchased separately) that can replace the main license key for a restricted period of time. It is possible to specify the backup license in the CHARON-VAX configuration file to prevent CHARON-VAX from stopping in case its main license is no longer accessible.

The CHARON-VAX license is read upon the start of each instance of CHARON-VAX and at a specified interval (defined by the license content) during the emulated system execution. If CHARON-VAX detects the absence (or malfunction) of the license key / software license, CHARON will try to use a backup license (if specified in the configuration file). If the license is not available / not specified, CHARON displays a warning message in the log file requesting license key reconnection or software license reactivation. If the license is not reconnected within a given period of time (the check interval), CHARON-VAX exits.

Note that if the time-restricted license is used and it expires, CHARON-VAX tries to find its replacement automatically and, if found, CHARON-VAX proceeds using the replacement license.

> The CHARON-VAX software license is not distributed in case of Proof-of-Concept and evaluation installations. Only hardware dongles are used in this case.
>
> It is important to connect the HASP license keys to the computer from time to time even if CHARON-VAX is not used. The keys contain a built-in accumulator that needs to be charged. If the accumulator is completely discharged, the license key can be fatally damaged.

Update of the CHARON-VAX license can be performed on the fly without stopping CHARON-VAX. At the next license check, CHARON-VAX will use the updated license normally.

The following sections list all the main parameters of the CHARON-VAX licensing mechanism.

Parameters defined by CHARON-VAX license

The following table represents all the parameters defined by CHARON-VAX license:

General	Products relevant	Optional
- Physical key ID - License Number - End user name - Master key ID - License release date and time - Update Number - Purchasing Company name. In most cases the company to which the key was issued originally	- Commercial product name - Commercial product code - Commercial product version and range of build numbers suitable for running - Range of CHARON-VAX virtual models available for running - Type of host CPU required - Host operating system required - Number of virtual CPUs enabled for virtual SMP systems - Minimum number of host CPU cores required - Minimum host memory required - Maximum memory emulated. If not present the value defaults to the maximum memory possible for the particular virtual system. Note that the maximum memory may not be available to the virtual system if the host computer has insufficient physical memory. - Maximum number of CHARON-VAX instances that can be run concurrently - Whether or not CHAPI (CHARON-VAX API) can be used with this product - Product and Field Test expiration dates (if any) - Product and Field Test executions counter (if any) - Maximum number of hosts that may run CHARON-VAX concurrently (in the case of a networking license) - Level of support (if any), end date of any support contract, the "First Line" Service Provider - Frequency of CHARON-VAX license checking during CHARON-VAX execution	- Possibility to attach hardware QBUS/UNIBUS hardware via adapter - Parameter that reduces the maximum speed of the program - Parameter that enables the product to support additional serial lines through an option board from a company such as DIGI - Parameter that prohibits use of Advanced CPU Emulation. If not present the Advanced CPU Emulation is enabled - Parameter that enables emulation of IEQ11-A IEEE488 Controller (on top of DCI-3100 IEEE488 Controller - Parameter that enables emulation of DRV11-WA I/O controller (on top of DCI-1100 I/O controller)

CHARON-VAX licensing models

CHARON-VAX licensing models are divided in 3 groups:

Regular Sentinel HASP keys

This is most common way of CHARON-VAX licensing, the CHARON-VAX license is embedded in a Sentinel HASP dongle. This license is available only on the host where the dongle is physically installed.

The CHARON-VAX installation procedure takes care of the Sentinel HASP run-time (driver) installation. Once the CHARON-VAX product has been installed, it is possible to plug-in the regular license key and proceed with CHARON-VAX usage without additional configuration steps.

> The number of CHARON-VAX instances allowed to run on a particular host may be restricted by the license content (see above).

Network Sentinel HASP keys

The Network Sentinel HASP key (red dongle) can be shared between several hosts running CHARON-VAX including the host on which the network license is installed.

If CHARON-VAX is installed on the host where the network key is connected, no additional steps are required. The Sentinel driver is activated as part of the CHARON-VAX installation. If the host does not have CHARON-VAX installed, the host can still distribute the connected network license to CHARON-VAX instances running on other hosts. In this case the Sentinel driver must be installed on the host manually.

The Sentinel run-time driver is distributed as a separate RPM package in the CHARON-VAX kit. Please see the "License installation" section of this chapter for details.

Once the Sentinel run-time driver is installed and the network license is connected, CHARON-VAX can be started on any appropriate host on the LAN network segment. In the current CHARON-AXP/VAX versions, a network license controls the maximum overall number of active instances, which can be distributed across client host systems according to the preference of the customer.

Software licenses

The CHARON-VAX Software License is a "virtual" key with exactly the same functionality as the hardware dongle.

The CHARON-VAX software license does not require any hardware but it requires the installation of the Sentinel run-time environment.

> Software licenses are best suited for stable environments, because their correct function depends on certain characteristics of the host system. Changing any of these characteristics will invalidate the license.
>
> - If the CHARON host runs on real hardware, the software licenses are by default **tightly bound to the hardware** for which they were issued. If major hardware characteristics of the system are changed, the license will be disabled.
> - If the CHARON host runs in a **virtual environment** (e.g. VMware), the software licenses are normally bound to the virtual machine ID and a set of additional characteristics of the virtual machine. If any of these parameters are changed, the license will be disabled.
>
> For a more detailed description of the restrictions, please refer to the Software Licensing restrictions article or contact your Stromasys representative.

Software licenses are always network-wide on Linux so they behave the same way as Network HASP keys.

Multiple licenses configuration and backup license

For any type of licensing, CHARON-VAX can use **only one valid ("active") license (of given vendor code) at a time**.

The "hasp_srm_view" utility displays the "active" license by default and is able to display all available licenses with the "-all" parameter. It is also possible to check some specific license by its number using the "-key" parameter.

The utility provides the license number and ID / IP address of the host where the active license is installed.

> CHARON-VAX **cannot**:
>
> - check all the available license keys / software licenses,
> - choose one,
> - automatically switch from one key to another (except when the backup license is specified in its configuration file).

The general recommendation is to avoid usage of multiple keys in one network segment. Use only one locally installed license per host or one network license per local network segment containing several CHARON-VAX hosts.

When needed, it is possible to use a special parameter in the CHARON-VAX configuration file to specify exactly which license must be used by each particular instance of CHARON-VAX:

Parameter	Type	Value
license_key_id[N], N=0 or 1	Numeric	A number (decimal Sentinel key ID) that specifies regular (N=0) and backup (N=1) license keys to be used by CHARON-VAX. **Example**: ``` set session license_key_id[0]=1877752571 set session license_key_id[1]=354850588 ``` It is also possible to specify both regular and backup key in one line. **Example**: ``` set session license_key_id[0]=1877752571 license_key_id[1]=354850588 ``` Depending on the presence of the regular and/or backup license key IDs in the configuration file, CHARON-VAX behaves differently: 1. **No keys are specified** CHARON-VAX behaves as usual (performs unqualified search for any suitable key). If no keys are found, CHARON-VAX exits. 2. **Both keys are specified** CHARON-VAX performs qualified search for regular license key. If it is not found, CHARON-VAX performs qualified search for backup license key. If it is not found, CHARON-VAX exits. 3. **Only regular key is specified** CHARON-VAX performs qualified search for regular license key. If it is not found, CHARON-VAX performs unqualified search for any suitable key. If it is not found, CHARON-VAX exits. 4. **Only backup key is specified** CHARON-VAX behaves as usual (performs unqualified search for any suitable key). If no keys are found, CHARON-VAX exits.

License installation

Installation of Regular and Network license keys

Installation of CHARON-VAX regular and network licenses consists of:

1. Installation of the Sentinel run-time environment on the CHARON-VAX host (regular and network keys) or on the host that will distribute CHARON-VAX licenses over a local network segment (network key only). The Sentinel software (the "aksusbd" RPM package) is installed automatically by CHARON-VAX for Linux.
2. Physical connection of the HASP license dongle to the CHARON-VAX host or to the host distributing the CHARON-VAX license over the local network segment.

When manual installation of Sentinel run-time is required (in the case of the network license server that does not have CHARON-VAX installed), open the CHARON-VAX kit folder and proceed the following way:

```
# rpm --nodeps -ihv aksusbd-2.5-1.i386.rpm charon-license-4.8-18302.el71.x86_64.rpm
```

In case of network-wide license (red dongle) do the following:

- *On the server side (where the network license will reside)*: open port 1947 for both TCP and UDP
- *On the client side*, if broadcast search for remote licenses is to be used, UDP traffic from port 1947 of the license server to ports 30000-65535 of the client must be permitted.
- *Both on server and client sides*: setup default gateway

Please consult with your Linux User's Guide on details.

If stricter firewall rules are required, it is possible to open the ports 30000-65535 and 1947 only for the "/usr/sbin/hasplmd" daemon.

> Some additional packages may be needed in certain cases, for example "glibc.i686"

Replacement of currently installed Sentinel run-time

Replacement of currently installed Sentinel Run-time may be needed in case of:
- Upgrade to a newer version of CHARON-VAX
- Installation of a specific CHARON-VAX license Run-time provided by STROMASYS

Run-time replacement is a two step process:
- Remove the current run-time (and the package "charon-license-<...>**.**rpm" containing the run-time customization) with the command

```
# rpm --nodeps -e aksusbd charon-license-<...>
```

- Change to the directory where the new run-time RPM resides (along with the corresponding "charon-license-<...>**.**rpm" customization package) a nd issue the command:

```
# rpm --nodeps -ihv aksusbd<…>.rpm charon-license-<...>.rpm
```

Installation and update of CHARON-VAX Software License or HL/HASP dongle License

CHARON-VAX Software Licenses (SL) can be installed / updated according to the procedure described below.
- Install CHARON-VAX together with Sentinel run-time (Sentinel run-time is an essential part of CHARON-VAX for Linux distribution)
- Reboot the host system
- In case of Software License installation and if there are already installed network-wide SL's on the local network, disable access to the network licenses in the following way:
    - Using your web browser, open the http://localhost:1947 page to access the "Sentinel HASP Admin Control Center" (ACC).
    - Select the "Configuration" option from the left panel then the "Access to Remote License Managers" tab.
    - Uncheck the highlighted options:

- Press the "Submit" button to apply settings
- Select the "Network" tab.
- Set the "Network visibility" to "None":

- Press "Submit" button to apply the settings.
- Do not forget to return these settings back after the SL installation.
- Connect the HASP dongle to the host system (in case of update of a license in a dongle)

- Collect the CHARON-VAX host fingerprint file (".c2v") - in case of first installation of a Software License:

```
# hasp_srm_view -fgp my_host.c2v
```

or collect the ".c2v" file in case a Software License is already installed or the connected HL/HASP dongle needs updating:

```
# hasp_srm_view -c2v current_license.c2v
```

- Send the ".c2v" file ("*my_host.c2v*" / "current_license.*c2v*" in the examples above) to STROMASYS
- Receive a ".v2c" file in return and put it somewhere on the CHARON-VAX host.
- Start any web browser on this system and go to http://localhost:1947 to access the "Sentinel HASP Admin Control Center" (ACC) or configure ACC for remote access (see the details below).
- In ACC, under the Options menu, select Update/Attach, "Browse" for the "*.v2c" file and then "Apply File".
- Ensure that the license appears in the "Sentinel Keys" menu.

Alternatively it is also possible to use the "hasp_update" utility for applying ".v2c" file.

The content of the installed software license is not shown by the Sentinel HASP Admin Control Center.To see it please run the "hasp_srm_view" utility from the local console or configure remote access according to the instructions given in the "hasp_srm_view" utility section.

In case of network-wide software license do the following:

- *On the server side (where the network license will reside)*: open port 1947 for both TCP and UDP
- *On the client side*, if broadcast search for remote licenses is to be used, UDP traffic from port 1947 of the license server to ports 30000-65535 of the client must be permitted.
- *Both on server and client sides*: setup default gateway

Please consult with your Linux User's Guide on details.

If stricter firewall rules are required, it is possible to open the ports 30000-65535 and 1947 only for the "/usr/sbin/hasplmd" daemon.

License management

CHARON-VAX license management is performed by the Sentinel Admin Control Center and specific utilities. They are described in the sub-sections below.

Sentinel Admin Control Center

General Description

The Sentinel Admin Control Center (ACC) is the web-interface to the Sentinel run-time environment. It allows viewing/managing available keys, enabling and disabling them, controlling usage of remote keys etc.

To access the ACC, start any web browser and open the http://localhost:1947 page.

The Sentinel Admin Control Center is not able to display CHARON-VAX licenses - to view key contents, use the "hasp_srm_view" utility.

To access the Sentinel Admin Control Center start any web browser and open the http://localhost:1947 page. The web interface of the Sentinel Admin Control Center will appear.

The screenshot below gives an example:

Sentinel Admin Control Center

Sentinel Keys Available

Options

Sentinel Keys
Products
Features
Sessions

Update/Attach

Access Log
Configuration
Diagnostics

Help
About

#	Location	Vendor	Key ID	Key Type	Configuration	Version	Sessions	Actions
1	XEON4WAYW7	68704	961833018	HASP HL NetTime 50	-	3.25	-	Browse / Net Features
2	Local	68704	354850588	HASP HL NetTime 50	-	3.25	-	Products / Features / Sessions / Blink on
3	Local	68704	1351199824	HASP HL Time	-	3.25	-	Products / Features / Sessions / Blink on
4	rh64	DEMOMA - evaluation	464243137687019632	HASP SL AdminMode Rehostable	-	2.31	1	Browse / Net Features

Details for HL NetTime 50 (ID:961833018) on 192.168.1.22
Key Hardware Version: 6.2
Sentinel License Manager Version: 12.50 Build 1.16926
Uptime: 7 days 23 hours 45 minutes
Host: XEON4WAYW7 running Windows 7 Ultimate Build 7601 Service Pack 1 (x86 Family 15 Model 2 Stepping 5)

This example demonstrates that 4 license keys are available:
1. A network key ("*HASP-HL NetTime*") on the host "XEON4WAYW7"
2. A network key installed locally
3. An HASP-HL installed locally
4. A network-wide software license on the host "RH64"

The Sentinel Admin Control Center reports that there is one opened session on key #4. The other keys are not being used at the moment.

Using the Sentinel Admin Control Center it is possible to check the available keys, verify the hosts on which they reside, verify the opened sessions, etc. For a more detailed description of the Sentinel Admin Control Center, please refer to its "Help" section.

Disable remote keys access

A helpful feature of the Sentinel Admin Control Center is the ability to disable access to remote keys. If the network key is installed locally, access to the key from remote hosts can be disabled. The following examples demonstrate how this can be done.

To disable access to remote keys, switch to the "Access to Remote License managers" tab, uncheck the "Allow Access to Remote Licenses" checkbox and press the "Submit" button to apply this setting:

Sentinel Admin Control Center

Configuration for Sentinel License Manager

Options

Sentinel Keys
Products
Features
Sessions

Update/Attach

Access Log
Configuration
Diagnostics

Help
About

| Basic Settings | Users | Access to Remote License Managers | Access from Remote Clients | Detachable Licenses |

Allow Access to Remote Licenses ☑ You may experience a delay of a few minutes before your changes will take effect.
Broadcast Search for Remote Licenses ☑
Aggressive Search for Remote Licenses ☐
Specify Search Parameters

Submit Cancel Set Defaults

To disable access to the locally installed license key from remote hosts, switch to the "Access from Remote Clients" tab, uncheck the "Allow Access from Remote Clients" checkbox and press the "Submit" button to apply this setting:

SafeNet.

Sentinel Admin Control Center

Configuration for Sentinel License Manager

Options
Sentinel Keys
Products
Features
Sessions
Update/Attach
Access Log
Configuration
Diagnostics
Help
About

| Basic Settings | Users | Access to Remote License Managers | Access from Remote Clients | Detachable Licenses |

Allow Access from Remote Clients   ☑ You may experience a delay of a few minutes before your changes will take effect.

Access Restrictions

```
allow=all
```

Show Recent Client Access   The entries are evaluated in the order in which they are specified. As soon as a match is found, evaluation stops.
allow=all is implicitly added to end of list

Submit   Cancel   Set Defaults

Accessing Sentinel Admin Control Center from remote hosts

By default, the Sentinel Admin Control Center forbids accessing its web interface from remote machines. To allow access, configure the ACC for remote management.

The first step is to edit the "`hasplm.ini`" file:

```
# vi /etc/hasplm/hasplm.ini
```

Allow remote access by changing the "ACCremote" parameter from "0" to "1" then restart the Sentinel Admin Control Center run-time:

```
# /etc/init.d/aksusbd restart
```

If the CHARON-VAX host firewall is blocking remote access to the Sentinel Admin Control Center, please configure the firewall to open the port 1947 (TCP protocol). Refer to the Linux documentation for details on how to configure the firewall. It is also possible to use SSH port forwarding with the following command (replace "*CHARON_MACHINE*" by the real CHARON-VAX host name):

```
# ssh -L8080:CHARON_MACHINE:1947 root@CHARON_MACHINE
```

This will expose the Sentinel Admin Control Center on port 8080 to any computer and it will believe commands are coming from the local host.

License management utilities

CHARON-VAX for Linux provides a specific utility for license management - "hasp_srm_view". This utility is used to display the license(s) content, to collect key(s) status information and host fingerprint (C2V) files.

> Applying updates (".v2c" files) is typically done using the Sentinel Admin Control Center (see above) but alternatively it is also possible to use the specific "hasp_update" utility.

Please refer to the Utilities section of this Guide for more details.

Removing CHARON-VAX software licenses

The following procedure must be applied to remove software license:

1. Using your web browser, open the http://localhost:1947 page to access the "Sentinel HASP Admin Control Center" (ACC).
2. In the "Sentinel HASP Admin Control Center" (ACC), locate the target "Sentinel SL AdminMode" license.
3. Press the "Certificates" button at the right side of the SL description:

4. Note the name of the corresponding certificate and path to the certificates base in the "Certificates" section.
5. Remove the target certificate file from the specified directory, in most cases: "`/var/hasplm/installed/68704/`".
6. Reboot the CHARON host.
7. Start the "Sentinel HASP Admin Control Center" (ACC) again to ensure that the SL has been removed.

License Deinstallation

To completely remove a CHARON-VAX license from a host, it is enough to remove the Sentinel run-time daemon (and the package "charon-license-<...>. rpm" containing the run-time customization) using the following command:

```
# rpm --nodeps -e aksusbd charon-license-<...>
```

Then just physically disconnect the license key (in the case of protection by dongles).

Special "backup" license keys

Backup keys are provided by STROMASYS along with standard license dongles. It is strongly recommended to order a backup key to recover immediately from damage or loss of the main license key. Backup keys use a counter (integer) value hardcoded inside the key. This integer value is a number of hours CHARON-VAX is allowed to run. Each time CHARON-VAX checks the license (every hour), the value is decreased (by 1 hour).Please note that backup keys have restricted functionality:
- CHARON run time is typically limited to 720 hours (30 days). This should be more than enough time to get a replacement from STROMASYS.
- A backup license may be valid only until a certain date. Please check with STROMASYS management.

CHARON-VAX for Linux utilities

CHARON-VAX provides the following set of utilities:

Utility	Description
mkdskcmd	Used to create CHARON virtual disk containers of custom or standard types. This utility also may be used to transfer virtual disks of one type to virtual disks of another type.
mtd	Used to create CHARON tape images from physical tapes and to write tape images back to physical tapes.
hasp_srm_view	Used to display the CHARON license contents, to collect the host system fingerprint and to transfer software licenses from one host to another.
hasp_update	Sentinel standard utility used to retrieve Sentinel protection key information, detache a license from a Sentinel SL key and rehost a license from a Sentinel SL key
ncu	Used to dedicate a host interface to CHARON-VAX, to release it back to the host and to manage CHARON virtual interfaces (TAPs).
CHARON Guest Utilities for OpenVMS	Used to manage virtual tapes and CHARON performance.

All these utilities (except for CHARON Guest Utilities for OpenVMS) are invoked from the Linux console command line.

mkdskcmd

Table of Contents

Description

The "mkdskcmd" utility:

- Creates empty disk images of a given standard disk type or a custom disk size
- Transfers existing disk images of one type to disk images of another type.

Creating disk images

The first step is to obtain the name of the disk that needs to be created:

```
$ mkdskcmd --list
```

This command results in a list of all supported disk types.

Choose the desired disk (for example "RZ22"), then use the "mkdskcmd" command to create the virtual disk image as shown below:

```
$ mkdskcmd --disk rz22 --output rz22.vdisk
```

A disk container "rz22.vdisk" will be created in the current directory.

> A file "rz22.avdisk" will also be created. This file helps CHARON accurately recognize a specific disk image type. It is recommended to put the ".avdisk" file in the same directory as the created disk image.

It is also possible to create custom disk images using "--blcount" (blocks count) and "--blsize" (blocks size) switches.

To get all the available parameters please use the "–help"switch:

```
mkdisk for CHARON utility v. 1.14
Copyright: STROMASYS, 2015

Usage:
mkdskcmd [Options]

Options:
--help - to see help screen
-h - to see help screen

--output <full name> - to specify output file name
-o <full name> - to specify output file name

--disk <disk name> - to specify the disk name from Disk table
-d <disk name> - to specify the disk name from Disk table

--blsize <number> - to specify the block size in bytes (custom disk image)
-z <number> - to specify the block size in bytes (custom disk image)

--blcount <number> - to specify number of the blocks (custom disk image)
-c <number> - to specify number of the blocks (custom disk image)

--avtable <full_name> - to specify AVDISK table file
-a <full_name> - to specify AVDISK table file

-t - please see the '--transform' options description
--transform <source_disk_name> <source_disk_params> - to transform the the disk image (change actual size)
<source_disk_name> - the file name of the disk image to be transformed
<source_disk_params> - the name of the disk from the list of available at the Disk table
The source disk size will changed accordingly the reach the specified parameters.
To specify the transform parameters manually, follow the option below:
--transform <source_disk_name> --blsize <number> --blcount <number>

--shrink - parameter which needs to be EXPLICITLY provided, if the disk size is to be decreased
-k - parameter which needs to be EXPLICITLY provided, if the disk size is to be decreased

--list <full_name> - to display AVDISK table
-l <full_name> - to display AVDISK table

--silent - silent mode running
-s - silent mode running

Return value:
0 - for Success
Non zero - in case of failure

Examples:
mkdskcmd -h
mkdskcmd -l
mkdskcmd -a /opt/charon/utils/mkdsk.vtable -o /etc/rk07.vdisk -d rk07
mkdskcmd -o /etc/custom.vdisk -z 512 -c 16384
mkdskcmd -t /etc/rz22.vdisk rz25 -a /opt/charon/utils/mkdsk.vtable
mkdskcmd -t /etc/rz22.vdisk rz25 -a /opt/charon/utils/mkdsk.vtable -z 512 -c 262134
```

The "--avtable" parameter is used to work with an alternative disk specification database (or to point to the standard database ("mkdsk.vtable") if it is in a location other than the current directory).

The "--blcount" (blocks count) and "--blsize" (blocks size) switches are used to create custom disk images.

Transferring disk images

The "mkdskcmd" utility is able to transfer (copy) disk images of one type to a disk image of another type.

This operation is needed, for example, to obtain more free space on a disk image that already contains data.

Note: it is not possible to add more free space dynamically. The virtual machine must be stopped before performing this operation.

> If a source disk image is larger than the target disk image, the extra data is lost. If the source disk image is smaller, it will be extended and padded with null bytes ('\0').

An example of the syntax follows:

```
$ mkdskcmd --transfer <source disk file name> <source disk parameters> [--shrink] [-k]
```

where:

- <source disk file name> - a file name of the disk image to be transferred
- <source disk parameters> - the name of the disk from the list provided by the "mkdskcmd --list" command execution or the disk geometry specification (see below).
- --shrink or -k - used in the case where the target disk is transferred to a smaller disk.

Example:

```
$ mkdskcmd --transfer /etc/rz22.vdisk rz25
```

It is also possible to specify the disk parameters manually with "--blcount / -c" (blocks count) and "--blsize / -z" (blocks size) switches:

```
$ mkdskcmd --transfer <source disk file name> -blsize <number> -blcount <number>
```

Example:

```
$ mkdskcmd -t /etc/custom.vdisk -z 512 -c 262134
```

> There is a certain delay between the moment when the utility reports that a disk image has been transferred and its actual availability to CHARON. This delay can reach to several minutes in case of very big disks transfers. It happens because the host operating systems needs some time for actual allocation of the enlarged file on HDD.

Back to Table of Contents

mtd

Table of Contents

Description

The "mtd" utility is used to:

- Create a CHARON tape image from a physical tape
- Write a tape image to a physical tape.

Usage is the following:

```
$ mtd [options] <tape device name> <tape container name>
```

Parameters:

```
MTD - CHARON Magnetic Tape Dump & Restore utility, Version 2.3 (Build 18302)
Copyright (C) 2009-2016 STROMASYS SA. All rights reserved.

Usage: mtd [options] <tape-drive-name> <file-name> - dump tape content to file
mtd <file-name> <tape-drive-name> - restore dump to tape
mtd <file-name> <file-name> - convert formats
mtd <file-name> - examine tape dump and check integrity

<tape-drive-name> - tape drive
<file-name> - name of tape container file (.mtd or .vtape)

Options:
-l <file-name> - log file name (.log)
-n - do not rewind tape
-r <number> - number of attempts to retry failing tape reads
-i - ignore failing tape reads (implies -r 0)
-p - disable progress reporting
-v - enable verbose trace of data transfer (implies -p)
-s - output tape image in SMA format
-g - gather statistics and print upon completion
```

Example:

```
$ mtd -l tape1.txt -r 10 /dev/st5 /charon/tapes/tape1.vtape
```

Tape container to physical tape transfer

Use the following syntax to write the content of a tape container to a physical tape:

```
$ mtd <tape container name> <tape device name>
```

Example:

```
$ mtd /charon/tapes/tape1.vtape /dev/st5
```

Back to Table of Contents

hasp_srm_view

Table of Contents

Description

The "hasp_srm_view" utility displays the CHARON licenses content.

Run the utility with one of the following parameters to see the license(-s) details:

- "-l" (or without parameters) - CHARON default license details
- "-all" - all available CHARON licenses details
- "-key <key number>" - specific CHARON license (defined by its "key number") details

The "hasp_srm_view" utility provides the following functionality:

- Display the licenses details. It is possible to view all available license or some specific one.
- Collecting license status information
- Collecting host fingerprint information

Run the utility without any options to display the license details.

```
# hasp_srm_view -help

CHARON Sentinel HASP utility
Copyright: STROMASYS, 2016

Options:
-? or -h or -help - to see help screen

-l - to see CHARON license details (for default key)
-all - to see CHARON license details (for all available keys)
-key <key number> - to see CHARON license details (for specific key)

-c2v <C2V file> - to collect the key status information (C2V file)
 -c2v <C2V file> -key <key number> - to collect C2V file for specific local key
-fgp <C2V file> - to collect the host fingerprint information (C2V file)
```

The specific type of CHARON license defines what switches may be used in each case.

Collecting the "c2v" file can be done only from the CHARON host console.

Remote collection of status information

For remote collection of status information it is recommended to use "ssh" as shown in the following examples:

```
# ssh root@CHARON_HOST /opt/charon/bin/hasp_srm_view -c2v /opt/charon/bin/my_hasp_key.c2v
# ssh root@CHARON_HOST /opt/charon/bin/hasp_srm_view -fgp /opt/charon/bin/my_host_fingerprint.c2v
```

To see the license text on the console:

```
# ssh root@localhost /opt/charon/bin/hasp_srm_view
```

To collect license text to an output file on host server:

```
# ssh root@localhost /opt/charon/bin/hasp_srm_view > /opt/charon/bin/hasp_srm_view.txt
```

The "hasp_srm_view" utility always reports the ID and IP address of the host(s) where active licenses are found.

Back to Table of Contents

hasp_update

Table of Contents

Description

The "hasp_update" is a Sentinel standard utility for license management included in the CHARON kit.

To invoke the "hasp_update" utility login as "root" and use the following syntax:

```
# hasp_update <option> <filename>
```

where:

Parameter	Value	Description
<option>	u	Updates a Sentinel protection key / attaches a detached license
	i	Retrieves Sentinel protection key information
	d	Detaches a license from a Sentinel Software License (SL) key
	r	Rehost a license from a Sentinel Software License (SL) key
	h	Display help
<filename>		Path to the V2C/H2R file when used with the 'u' option
		Optional path to the C2V file when used with the 'i' option Uses "stdout" if file name is not specified

Example:

```
# hasp_update u license_update.v2c
```

Usage

We recommend to use this tool only for "Update a Sentinel protection key / attach a detached license" function ("u" option). For the rest use "hasp_srm_view" utility.

Back to Table of Contents

ncu

Table of Contents

Description

The "ncu" ("Network Control Utility") is used to dedicate a host interface to CHARON, to release it back to the host and to manage CHARON virtual interfaces (TAPs).

The utility allocates chosen network interfaces (both physical and virtual) and configures the offload parameters.

> The NetworkManager service must be running in order to have all the functionalities enabled with "ncu". If the service is not enabled please configure the network using the manual operations described in the section "Manual configuration of CHARON networking" of the Installation chapter of this Guide.

Dedication of a host physical interface to CHARON

Login as root and enter "ncu". The following menu will appear:

```
# ncu
CHARON Network Configuration Utility, STROMASYS (c) 2016 Version 1.6

Interfaces Dedicated to State
---------- ------------ ------------
eth0       host         connected to host
eth1       host         connected from host
lo         host         unmanaged from host


================================================================
bridge name        bridge id        STP enabled        interfaces
========================= VLAN ================================
================================================================

select action:

1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit

:> 8
```

The utility lists available network interfaces (both physical and virtual) and indicates whether they are dedicated to the host or to CHARON and whether they are currently in use by host operating system.

"ncu" offers several options:

1. Dedicate interface to CHARON
2. Release interface to host
3. Create a bridge between a chosen physical network interface and the Linux virtual network and create a number of virtual network interfaces
4. Remove the Linux virtual network and all the created virtual network interfaces
5. Add VLAN interface
6. Remove VLAN interface
7. Print status - use it to display status of network interfaces and the menu shown above
8. Exit

In the example above we see 2 network interfaces - "eth0" and "eth1", both of them are dedicated to host, but host uses only the interface "eth0".

Let's dedicate the interface "eth1" to CHARON. Enter "1", type "eth1" and press Enter:

```
Specify the interface to dedicate to CHARON:eth1
Turning off offloading for eth1.. Please wait

select action:

1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit

:> 7
```

Now the interface "eth1" is dedicated to CHARON:

```
Interfaces Dedicated to State
---------- ------------ ------------
eth0       host         connected to host
eth1       CHARON       disconnected from host
lo         host         unmanaged from host

================================================================
bridge name            bridge id         STP enabled     interfaces

========================= VLAN =================================
================================================================

select action:

1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit
```

Enter "8" to return to console prompt.

Now "eth1" can be used by CHARON.

Release of a host physical interface back to host

Login as root and enter "ncu". The following menu will appear:

```
# ncu
CHARON Network Configuration Utility, STROMASYS (c) 2016 Version 1.6

Interfaces Dedicated to State
---------- ------------ ------------
eth0       host         connected to host
eth1       CHARON       disconnected from host
lo         host         unmanaged from host


================================================================
bridge name           bridge id        STP enabled      interfaces
======================== VLAN ===============================
================================================================

select action:

1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit

:> 2
```

Let's say that we want to return the interface "eth1" (currently dedicated to CHARON) back to host. To do that enter "2" then "eth1":

```
Specify the interface to release to HOST:eth1
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/3)

select action:

1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit

:> 8
```

Enter "8" to quit the "ncu" utility.

The interface "eth1" is released back to host system now.

[Back to Table of Contents](#)

Creation of a virtual network

Login as root and enter "ncu":

```
# ncu
CHARON Network Configuration Utility, STROMASYS (c) 2016 Version 1.6

Interfaces Dedicated to State
---------- ----------- -----------
eth0       host        connected to host
eth1       host        connected to host
lo         host        unmanaged from host


================================================================
bridge name         bridge id          STP enabled     interfaces
======================== VLAN =================================
================================================================

select action:

1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit

:> 3
```

Enter "3" to create a bridge between the host physical network adapter and the LINUX virtual network interfaces (TAP) and specify the physical network interface ("eth1" in our example) and the number of virtual network interfaces to be created (2 in our example):

```
Specify the interface to be used for BRIDGE:eth1
How many tap should be created:2
Forming the bridge: ..1..2..3..4..5.. addif tap0 .. addif tap1 ..7..8 done!
Formed bridge br0_eth1 attached over eth1...

select action:

1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit


:> 7
```

Now enter "7" to see the created virtual interfaces:

```
Interfaces    Dedicated to    State
----------    ------------    ------------

eth0          host            connected to host
eth1          bridge          connected to bridge
lo            host            unmanaged from host
tap0          CHARON          connected to host
tap1          bridge          connected to bridge
===============================================================
bridge name          bridge id            STP enabled        interfaces

br0_eth1             8000.768e1ea091d9   no                 eth1
tap0
tap1
======================== VLAN =================================
===============================================================

select action:

1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit

:> 8
```

In the example above we see 2 virtual network Interfaces "tap0" and "tap1" connected to the created bridge. The physical network interface "eth1" is used for the bridge to the virtual network interfaces.

The interfaces "tap0" and "tap1" are ready to be used in CHARON configurations - they do not need to be additionally dedicated to CHARON.

Enter "8" to quit "ncu" utility.

Back to Table of Contents

Removal of a virtual network

Login a root. Start "ncu" utility:

```
# ncu

CHARON Network Configuration Utility, STROMASYS (c) 2016 Version 1.6

Interfaces    Dedicated to     State
----------    ------------     ------------

eth0          host             connected to host
eth1          bridge           connected to bridge
lo            host             unmanaged from host
tap0          CHARON           connected to host
tap1          bridge           connected to bridge
================================================================
bridge name        bridge id          STP enabled       interfaces

br0_eth1           8000.768e1ea091d9  no                eth1
tap0
tap1
========================= VLAN =================================
================================================================

select action:

1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit

:> 4
```

Enter "4" then enter the interface name that is a bridge to the Linux virtual network on this host ("eth1" in our example):

```
Specify the phys interface used for BRIDGE:eth1
Cleanup bridge br0_eth1 with ip over eth1...
Removing the bridge: ..1..2 delif eth1
delif tap0
delif tap1
..5..6..7..8 done!

select action:

1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit

:> 8
```

Enter "8" to quit "ncu" utility.

Adding VLAN interface

Login a root. Start "ncu" utility:

```
# ncu

CHARON Network Configuration Utility, STROMASYS (c) 2016 Version 1.6

Interfaces    Dedicated to    State
----------    ------------    ------------

eth0      host          connected to host
eth1      host          connected to host
lo        host          unmanaged from host


================================================================
bridge name          bridge id          STP enabled      interfaces
======================= VLAN ================================
================================================================

select action:

1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit

:> 5
```

Enter "5" then enter:

1. The phisical interface name to be used for creating VLAN
2. The ID of the VLAN device
3. IP address of the VLAN device. Skip this step if no IP is required
4. Network mask of the VLAN device. Enter for no network mask.

```
Specify the phys interface used for VLAN:eth1
Specify the id of VLAN device (<4095):111
Specify the ip address of VLAN device or empty string for no ip address: 192.168.1.100
Specify the netmask address of VLAN device or empty string for no netmask:
225.225.225.0

select action:

1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit

:> 8
```

Enter "8" to quit "ncu" utility.

Removing VLAN interface

Login a root. Start "ncu" utility:

```
# ncu

CHARON Network Configuration Utility, STROMASYS (c) 2016 Version 1.6

Interfaces    Dedicated to    State
----------    ------------    ------------

eth0      host         connected to host
eth1      host         connected to host
lo        host         unmanaged from host


================================================================
bridge name         bridge id         STP enabled    interfaces

======================= VLAN ================================
eth1.111
================================================================

select action:

1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit

:> 6
```

Enter "6" then enter the VLAN interface for remove:

```
Specify the VLAN interface, which be removed: eth1.111

Removed VLAN -:eth1.111:-

select action:

1 - Dedicate to CHARON
2 - Release to host
3 - Create Bridge with TAPs
4 - Remove Bridge
5 - Add VLAN
6 - Remove VLAN
7 - Print status
8 - Exit

:> 8
```

Enter "8" to quit "ncu" utility.

CHARON Guest Utilities for OpenVMS

Table of Contents

Description

The "CHARON Guest Utilities for OpenVMS" (CHARONCP) package contains several utilities for managing virtual tapes, changing the emulator speed and creating useful definitions for that operations.

This set of utilities is located in the "charoncp011.vdisk" disk file in the "/opt/charon/disks" folder.

Supported OpenVMS versions (depending on platform support): OpenVMS 6.1 and above.

ⓘ In case of OpenVMS upgrade, CHARONCP will have to be re-installed.

Installation

Specify this image in the CHARON configuration file, boot from the system disk and mount the disk with the following OpenVMS command:

```
$ MOUNT <device name> /OVERRIDE=IDENTIFICATION
```

Issue the following commands to install the package (example given for OpenVMS V8.4):

```
$ @SYS$UPDATE:VMSINSTAL

...

* Are you satisfied with the backup of your system disk [YES]? YES
* Where will the distribution volumes be mounted: <device name>:[CHARONCP011.KIT]

Enter the products to be processed from the first distribution volume set.
* Products: CHARONCP011

* Enter installation options you wish to use (none): <press enter>

...

Do you want to install this product [NO]? YES

...

* Where should the CHARONCP root directory be located ? [SYS$SYSDEVICE:[CHARONCP]]: <press enter>

...

* Do you want to purge files replaced by this installation [YES]? <press enter>
```

Select all the components included to the package:

```
                     Component Selection

Select the CHARONCP components you wish to install from the menu below.
An asterisk appears next to the packages that have already been
selected. You can remove a package from the list by selecting it
again. You may enter more than one selection by separating your
choices with commas.

1. [*] CHARONCP Guest Utility (REQUIRED)
2. [*] Compatability Utilities
3. [*] Install DCL Commands & Help

4. Exit

* Your choice [4]: 1,2,3

...

* Your choice [4]: 4

...

* Is this correct [YES]: <press enter>

...

* Products: <press enter>

        VMSINSTAL procedure done at hh:mn

$
```

Proceed with installation using all the default options.

Once the installation is completed, add the following line to the "SYS$STARTUP:SYSTARTUP_VMS.COM" ("SYS$STARTUP:SYSTARTUP_V5.COM" for VMS 5.5) file for the package to be loaded automatically at system startup::

```
$ @SYS$STARTUP:CHARONCP_STARTUP
```

After that the package will be loaded automatically on startup.

Performance optimization

CHARON takes 100% of host CPU even in case of idle state of guest OpenVMS operating system. To get rid of such resources consumption there is a specific option provided by CHARON Guest Utilities - "idle" mode.

`$ CHARONCP SET IDLE /ENABLE`	Loads the OpenVMS idle loop detection software. This allows CHARON to detect when the emulated CPU(s) are idle and use the host power saving instructions to reduce power usage.
`$ CHARONCP SET IDLE /DISABLE`	Unloads the OpenVMS idle loop detection software.

Adjusting emulator's speed

CHARON speed can be adjusted using these commands:

`$ ` **`CHARONCP SET SPEED /UP=`**`<number of steps>`	Increase the emulator speed by the specified number of steps.
`$ ` **`CHARONCP SET SPEED /DOWN=`**`<number of steps>`	Decrease the emulator speed by the specified number of steps.
`$ ` **`CHARONCP SET SPEED /RESET`**	Remove all speed stepping restraints.

Emulator's shutdown control

It is possible to schedule CHARON to shutdown directly from OpenVMS:

`$ ` **`CHARONCP SET SHUTDOWN /IN=`**`<seconds, 60 is default>`	The number of seconds before the CHARON emulator is to shut down. The value can be between 0 and 65535 (approx. 18 hours).
`$ ` **`CHARONCP SET SHUTDOWN /RESET`**	Clear a scheduled shutdown.

Virtual tapes management

Specify mapping to tape container in the following way in the CHARON configuration file:

```
set <adapter name> container[<unit name>] = ".vtape" removable[<unit name>] = true
```

Example:

```
set PKA container[600] = ".vtape" removable[600] = true
```

Once it is done using the following commands it is possible to manage virtual tapes attached to CHARON:

`$ ` **`CHARONCP SET MAGTAPE`** `<device>` **`/LOAD=`**`"<filename>`**`.vtape"`**	Create the specified host-file (if it does not already exist) and attach it to the specified virtual tape device. **Example**: `$ ` **`CHARONCP SET MAGTAPE MKA600: /LOAD="backup_01.vtape"`**
`$ ` **`CHARONCP SET MAGTAPE`** `<device>` **`/UNLOAD`**	Detach any file currently attached to te specified virtual tape device. **Example**: `$ ` **`CHARONCP SET MAGTAPE MKA600: /UNLOAD`**

Possible errors:

BADFILENAME	The filename specified as a value to the qualifier /LOAD was either too long or does not have a file extension of ".vtape".
DEVNOTDISM	Attempting to execute a SET MAGTAPE/LOAD when a file is already attached. Perform a SET MAGTAPE/UNLOAD first. If a SET MAGTAPE/LOAD command has not previously been executed, then the CHARON configuration container specification for the tape device may contain a full path. Doing this will create and attach and initial tape container file. To avoid this, remove the file name from the specification (leaving only a fileextension of ".vtape" and optional directory).

ℹ️ If some tape container has been already specified in the CHARON configuratioon file use the command "CHARONCP SET MAGTAPE <device> /UNLOAD" to unload it first.

Defining keys

It is possible to define certain keys on the terminal keyboard for fast access to the CHARONCP functionalilty while you are in CHARONCP:

``` $ CHARONCP CHARONCP> DEFINE /KEY <key-name > <equivalence-string> ```	Defines an equivalence string and a set of attributes with a key on the terminal keyboard. You can have a set of keys defined automatically for use with the CHARONCP utility by placing DEFINE/KEY commands in the file SYS$LOGIN:CHARONCP_KEYDEFS.INI **Example**: ``` $ CHARONCP CHARONCP> DEFINE /KEY F1 "SET MAGTAPE MKA600: /UNLOAD" ```
``` $ CHARONCP CHARONCP> SHOW KEY <key-name> ```	Displays key definitions created with the DEFINE/KEY command. Refer to the DCL help entry for SHOW KEY for further information. **Example**: ``` $ CHARONCP CHARONCP> SHOW KEY F1 DEFAULT key state definitions: F1 = "set magtape mka600: /unload" CHARONCP> ```

ℹ️ For more information refer to the OpenVMS DCL Dictionary (DEFINE/KEY section).

Displaying version

``` $ CHARONCP SHOW VERSION ```	Displays the CHARONCP package version number and architecture. This can be useful for customers reporting issues with the CHARONCP software. **Example**: ``` $ CHARONCP SHOW VERSION CHARONCP version id is: V1.1 ```

CHARON-VAX for Linux configuration details

Introduction

This chapter describes in detail all configuration parameters of the devices emulated by CHARON-VAX for Linux, with corresponding examples of their loading and parameters.

Emulated devices are loaded with "load" command (if device has not been already loaded) and parameters are made active the "set"command. It is possible to specify parameters directly in the "load" command.

Example:

```
load RQDX3/RQDX3 DUA
set DUA container[0]="/charon/disks/user.vdisk"
```

In this example, an instance of an RQDX3 controller is loaded with a name "DUA". Its first unit ("container[0]") is mapped to "/charon/disks/user.vdisk" disk image.

> The Controller name is accompanied with a "/<module name>". The module name is a CHARON-VAX component that specifies the controller load module. Its name can be the same as the loaded controller, however this is not mandatory. Once a module name is specified there is no need to specify it again for additional references of the same controller..

Details of CHARON-VAX configuration

- General Settings
- Core Devices
- Serial lines
- Disks and tapes
  - MSCP and TMSCP Controllers
  - SCSI Controllers
  - DSSI Subsystem
  - CI Subsystem
  - Finding the target "/dev/sg" device
- Networking
- Sample configuration files
  - VAX 4000 Model 108 configuration file
  - VAX 6310 configuration file
  - VAX 6610 configuration file

Back to Table of Contents

General Settings

Table of Contents

Session

General settings that control execution of CHARON-VAX belong to an object callled "session". It is a preloaded object; therefore, only "set" commands apply.

Example:

```
set session <parameter>=<value>
```

The following tables describe all available "session" parameters, their meaning and examples of usage:

hw_model

Parameter	hw_model
Type	Text string
Value	Virtual VAX system hardware model to be emulated.

Use a default configuration template for each particular model as a starting point for a custom configuration. This would ensure that the parameter is set correctly.

Example:

```
set session hw_model="VAX_6610"
```

The models are:
- MicroVAX_3100_Model_96
- MicroVAX_3100_Model_98
- MicroVAX_3600
- MicroVAX_3900
- MicroVAX_II
- VAXserver_3600
- VAXserver_3600
- VAXserver_3600_128
- VAXserver_3600_512
- VAXserver_3900
- VAXserver_3900_128
- VAXserver_3900_512
- VAX_4000_Model_106
- VAX_4000_Model_108
- VAX_4000_Model_700
- VAX_4000_Model_705
- VAX_6000_Model_310
- VAXstation_4000_Model_90
- VAX_6610
- VAX_6620
- VAX_6630
- VAX_6640
- VAX_6650
- VAX_6660

configuration_name

Parameter	configuration_name
Type	Text string
Value	Name of CHARON-VAX instance (unique):

```
set session configuration_name="MSCDV1"
```

The value of this parameter is used as prefix to the event log file name when multiple log files are configured (see below).

From the example above, the CHARON-VAX log file will have the following name:

```
MSCDV1-YYYY-MM-DD-hh-mm-ss-xxxxxxxxx.log
```

xxxxxxxxx is an increasing decimal number starting from 000000000 to separate log files with the same time of creation (in case the log is being written faster than one log file per second).

log	
Parameter	log
Type	Text string
Value	The log file or directory name is where the log file for each CHARON-VAX execution session is stored.

Log specified as a file name

It is possible to overwrite the existing log file or to extend it using the "log_method" parameter.

ⓘ The "log_method" parameter is effective only when a single log file is specified, not a directory.

Example:

```
set session log="/charon/vax4106prod.log"
```

Log specified as a directory

CHARON-VAX automatically creates individual log files for each CHARON-VAX execution session. If the log parameter is omitted, CHARON-VAX creates a log file for each CHARON-VAX execution session in the directory where the emulator was started. In these two cases, the log rotation mode is enabled, meaning a new log file is created each time the virtual machine is started and when the log file size exceeds the one specified (see log_file_size) and/or when the log file is older than a specified number of days (see log_rotation_period).

ⓘ A symbolic link located in the same directory will be created, pointing to the active log file. Its name is based on the hw_model parameter or the configuration_name parameter if specified.

If the "configuration_name" parameter of the session is specified, the log file name is composed as follows:

```
<configuration_name>-YYYY-MM-DD-hh-mm-ss-xxxxxxxxx.log
```

If the "configuration_name" parameter is omitted, the log file name will have the following format:

```
<hw_model>-YYYY-MM-DD-hh-mm-ss-xxxxxxxxx.log
```

where "xxxxxxxxx" is an increasing decimal integer, starting from 000000000 to separate log files with the same time of creation (in case the log is being created faster than one log file per second).

⚠ Only existing directory can be specified. If the directory specified does not exist, this will be considered as a flat file.

Example:

```
set session configuration_name="vax4106prod"

set session log="/charon/logs"
```

The execution of the virtual machine will create a log file, named `/charon/logs/vax4106prod-2016-10-13-10-00-00-000000000.log` (for example) and a symbolic link named `/charon/logs/vax4106prod.log` pointing to this file. The link will be updated when the log rotation will occur.

log_method

Parameter	log_method
Type	Text string
Value	• "overwrite" (default) • "append" Determines if previous log information is maintained. > This parameter must be specified only in addition to "log" parameter on the same line. This parameter is applicable only if the CHARON-VAX log is stored in a file that is specified explicitly with the "log" parameter. **Example**: `set session log="log.txt" log_method="append"`

log_file_size

Parameter	log_file_size
Type	Text string

Value	If log rotation is enabled, the log_file_size parameter determines the log file size threshold at which the log is automatically rotated.
	• "unlimited" or "0" (default) - the feature is disabled
	• "default" - default size is used (4Mb)
	• <size>[KMG] - size of the current log file in bytes with additional multipliers:
	   • K - Kilobyte - multiply by 1024
	   • M - Megabyte - multiply by 1024*1024
	   • G - Gigabyte - multiply by 1024*1024*1024
	
	Examples:
	
	`set session log_file_size="default"`
	
	`set session log_file_size=10M`
	
	⚠️ Minimum LOG File size is 64K, maximum is 1G. Setting size less then 64K effectively makes the LOG File unlimited.

log_rotation_period

Parameter	log_rotation_period
Type	Text string

Value	• "default" - default value, 7 days. This values is used even if the "log_rotation_period" is not specified.
	• "daily" or "1"
	• "weekly" or "7"
	• "never" or "0"
	• <N> - in N days
	
	If the rotation log mode is enabled this parameter controls switching to next log file based on period of time passed. If enabled the switching to next log file appears at midnight.
	
	Examples:
	
	`set session log_rotation_period="weekly"`
	
	`set session log_rotation_period=14`
	
	ℹ️ If enabled the switching to next log file appears at midnight

license_key_id

Parameter	license_key_id[N] N=0 or 1
Type	Numeric
Value	A number (decimal Sentinel key ID) that specifies regular (N=0) and backup (N=1) license keys to be used by CHARON-VAX.

```
set session license_key_id[0]=1877752571
set session license_key_id[1]=354850588
```

It is also possible to specify both regular and backup key in one line:

```
set session license_key_id[0]=1877752571 license_key_id[1]=354850588
```

Based on the presence of the regular and/or backup license key IDs in the configuration file, CHARON-VAX behaves as follows:

1. **No keys are specified**
   CHARON-VAX performs an unqualified search for any suitable key. If no key is found, CHARON-VAX exits.
2. **Both keys are specified**
   CHARON-VAX performs a qualified search for a regular license key. If it is not found, CHARON-VAX performs a qualified search for backup license key. If it is not found, CHARON-VAX exits.
3. **Only regular key is specified**
   CHARON-VAX performs a qualified search for a regular license key. If it is not found, CHARON-VAX performs an unqualified search for any suitable key. If none are found, CHARON-VAX exits.
4. **Only backup key is specified**
   CHARON-VAX performs an unqualified search for any suitable key. If no key is found, CHARON-VAX exits.

license_key_lookup_retry

Parameter	license_key_lookup_retry
Type	Text String
Value	In case the CHARON-VAX license connection is not present at guest statup, this parameter specifies how many times CHARON-VAX will try to reestablish the connection and, optionally, a period of time between retries.

Syntax:

```
set session license_key_lookup_retry = "N [, T]"
```

where:

- N - Number of retries looking for license key (or keys)
- T - Time between retries in seconds. If not specified 60 seconds is used

Example 1

```
set session license_key_lookup_retry = 1
```

If license key is not found during initial scan, do only one more attempt after 60 seconds.

Example 2

```
set session license_key_lookup_retry = "1,30"
```

Same as above but retry in 30 seconds.

Example 3

```
set session license_key_lookup_retry = "3,10"
```

If license key not found during initial scan, do 3 more attempts waiting 10 seconds between them.

Example 4

```
set session license_key_lookup_retry = "5"
```

If license key is not found during initial scan, do 5 more attempts waiting 60 seconds between them.

affinity

Parameter	affinity
Type	Text string
Value	Overrides any initial process affinity mask provided by the host operating system. Once specified it binds the running instance of the emulator to particular host CPUs.Can be used for soft partitioning host CPU resources and/or for isolating host CPUs for other applications.By default CHARON-VAX emulator instance allocates as many host CPUs as possible. The "affinity" parameter overrides that and allows explicit specification on which host CPU the instance must run.

> The "affinity" parameter defines the total number of host CPUs to be used both for emulated VAX CPUs and for CHARON-VAX / CHARON-PDP application itself (including the CPUs to be used for I/O - they are controlled by "n_of_io_cpus" parameter described below).

Host CPUs are enumerated as a comma separated list of host system assigned CPU numbers:

```
set session affinity="0, 2, 4, 6"
```

n_of_io_cpus

Parameter	n_of_io_cpus
Type	Numeric
Value	This parameter specifies how many host CPUs CHARON-VAX must use for I/O handling. Use of the "affinity" parameter may limit the number of CPUs available.By default the CHARON-VAX instance reserves one third of available host CPUs for I/O processing (round down, at least one). The "n_of_io_cpus" parameter overrides that by specifying the number of CHARON I/O CPUs explicitly. **Example**: `set session n_of_io_cpus=2`

File inclusion

It is possible to include a configuration file into an existing one using the `"include"` command. File extension is usually `.icfg`.

Format:

```
include "file.icfg"
```

Example:

```
include "/charon/commonpart.icfg"
```

Core Devices

Table of Contents

CPU

ace_mode

The CHARON-VAX emulated CPU is configured with the "ace_mode" parameter.

Two VAX CPU implementations are available: the standard VAX instruction decoder and the optional high performance Advanced CPU Emulation mode ("ACE"). The ACE option optimizes VAX instruction interpretation and significantly improves performance. It also requires approximately twice the host memory to store the optimized code.

ACE optimization is performed dynamically during execution. Since it does not need to write optimized code back to disk, ACE provides its full capability instantly. The optimization does not compromise VAX instruction decoding; CHARON-VAX remains fully VAX hardware compatible and completely transparent to the VAX operating systems and applications.

Both CPU implementations passed the HP VAX Architecture (AXE) tests, the standard qualification for VAX instruction execution correctness.

The default VAX CPU mode is determined by the specific CHARON-VAX product license.

Parameter	ace_mode
Type	Boolean
Value	true or false.

This statement enables ACE mode if the CHARON-VAX license permits it. If this statement is omitted from the CHARON-VAX configuration file and the license permits it, "true" is the default. Otherwise "false" is the default. For test purposes the ACE mechanism can be disabled with:

```
set cpu ace_mode=false
```

> `"set cpu ace_mode=true"` is ignored when the license does not permit ACE operation.

The CHARON-VAX log file displays the status of the ACE option. ACE mode is disabled when the host system does not meet the minimum physical requirements for operation. If the emulator appears not to run at its normal performance, check the log file for a change in ACE mode and verify that sufficient host resources, especially memory, are available.

Back to Table of Contents

RAM

The CHARON-VAX memory subsystem is permanently loaded and has the logical name "ram".

size

Parameter	size
Type	Numeric
Value	Size of emulated memory in MB.

Example:

```
set ram size = 512
```

The amount of memory is capped at a maximum, defined in the CHARON license key. If the host system cannot allocate enough memory to map the requested emulated memory, CHARON-VAX generates an error message in the log file and reduces its effective memory size.

The following table lists the values of emulated RAM for various hardware models of virtual VAX systems:

Hardware Model	RAM size (in MB)			
	Min	Max	Default	Increment
MicroVAX_II	1	16	16	1,8,16
MicroVAX_3600	16	64	16	16
MicroVAX_3900	16	64	16	16
VAXserver_3600	16	64	16	16
VAXserver_3900	16	64	16	16
VAXserver_3600_128	32	128	32	32
VAXserver_3900_128	32	128	32	32
MicroVAX_3100_Model_96	16	128	16	16
VAXstation_4000_Model_90	16	128	16	16
VAX_4000_Model_106	16	128	16	16
VAX_6000_Model_310	32	512	32	32
VAXserver_3600_512	32	512	32	32
VAXserver_3900_512	32	512	32	32
MicroVAX_3100_Model_98	16	512	16	16
VAX_4000_Model_108	16	512	16	16
VAX_4000_Model_700	64	512	64	64
VAX_4000_Model_705	64	512	64	64
VAX_6610	128	3584	128	128
VAX_6620	128	3584	128	128
VAX_6630	128	3584	128	128
VAX_6640	128	3584	128	128
VAX_6650	128	3584	128	128
VAX_6660	128	3584	128	128

TOY

CHARON-VAX maintains its time and date via the "toy" (time-of-year) component. In order to preserve time and date while a virtual system is not running, the TOY component uses a binary file on the host system to store date and time relevant data. The name of the file is specified by the "container" option of the "toy" component.

container

Parameter	container
Type	Text string
Value	Specifies a name of a file in which CHARON-VAX preserves time and date during its "offline" period. This file also keeps some console parameters (such as default boot device). By default it is left unspecified. ℹ️ It is recommended to specify the full path to the file

Example:

```
set toy container="/charon/my_virtual_system.dat"
```

The CHARON-VAX time zone may be different from that of the host system. Correct CHARON time relies on the correctness of the host system time to calculate the duration of any CHARON "offline" periods. (i.e. while virtual system is not running). Every time CHARON comes on line it calculates a Delta time (the system time is used if there is no TOY file). Therefore, if the host system time is changed while CHARON is not running, the CHARON time may be incorrect when CHARON is restarted and the CHARON time must be set manually.

ROM

The System Flash ROM file conserves specific parameters between reboots.

container

Parameter	container
Type	Text string
Value	Specifies the name of a file in which CHARON-VAX stores an intermediate state of its Flash ROM. This state includes, for example, most of the console parameters. By default it is left unspecified. ℹ️ It is recommended to specify the full path to the file

Example:

```
set rom container="/charon/my_virtual_system.rom"
```

EEPROM

container

Parameter	container
Type	Text string
Value	A string specifying a file name to store content of EEPROM. **Example**: ```set eeprom container="vx6k610.rom"``` This command enables EEPROM parameters (e.g., default boot drive) to be automatically saved to a specified file. The EEPROM file is created in the directory in which CHARON-VAX starts and is created or overwritten each time any parameter relevant to EEPROM content is changed. ⓘ It is recommended to specify the full path to the file

Example:

```
set eeprom container="/charon/my_virtual_system.rom"
```

Auto boot

CHARON-VAX systems may be configured to boot the operating system automatically at start up.

MicroVAX3100, VAXstation 4000, VAX6310 and VAX 4000

Those models boot automatically if correct boot flags are set (and saved in the VAX console files) using the following command:

halt

Console Parameter	halt
Description	Determines whether the MicroVAX3100, VAXstation 4000, VAX6310 and VAX 4000 boot automatically if correct boot flags are set (and saved in the VAX console files). The value is: • *"reboot"* **Example**: ```>>>set halt reboot```

Please check that the "toy container" and "rom container" parameters are specified in the configuration file to store the boot flags.

MicroVAX II, MicroVAX 3600/3900 and VAXserver 3600/3900

The ROM of the MicroVAX II, MicroVAX 3600, MicroVAX 3900, VAXserver 3600 and VAXserver 3900 does not allow the VAX console to accept the command setting "auto-boot". Instead, automatic boot on startup can be specified in the CHARON-VAX configuration file as follows:

bdr boot

Parameter	bdr boot
Type	Text string
Value	Determines whether the MicroVAX II, MicroVAX 3600, MicroVAX 3900, VAXserver 3600 and VAXserver 3900 boot automatically if correct boot flags are set (and saved in the VAX console files).The value is: • `"auto"` **Example**: <pre>set bdr boot=auto</pre>

Please check that the "toy container" and "rom container" parameters are specified in the configuration file to store the boot flags.

VAX66x0

xmi boot

Parameter	xmi boot
Type	Text string
Value	Determines whether the CHARON VAX66x0 startup procedure stops at the ">>>" prompt after self-tests. The values are: • `"auto"` • `"manual"` (default) **Example**: <pre>set xmi boot = "auto"</pre> The value `"auto"` enables automatic boot from a default boot specification previously configured in the VAX console.The value `"manual"` disables automatic boot once the self tests are passed.

Please check that the "toy container" and "rom container" parameters are specified in the configuration file to store the boot flags.

Serial lines

Table of Contents

General Description

Configuration of CHARON-VAX serial lines is performed in 3 steps:

1. Loading virtual serial lines controller, for example:

```
load DHV11/DHV11 TXA
```

In this example, an instance of a "DHV11" serial line controller is loaded and named "TXA"

Note that VAX console adapters ("UART", "QUART") do not need to be loaded; they are preloaded.

2. Mapping an object type to host resources, for example:

```
load physical_serial_line TTA1
set OPA0 line="/dev/ttyS1"
```

In this example the object "physical_serial_line" is loaded, named "TTA1", and mapped to the "/dev/ttyS1" host physical serial port.

3. Connect the loaded virtual line controller and the mapped object:

```
set TXA line[5]=TTA1
```

In this example, the 6th line of the DHV11 controller "TXA" loaded at step 1 is connected to the mapping object "TTA1" loaded at the step 2.

Back to Table of Contents

Console

CHARON-VAX offers a one- or four-port serial console depending on the specified VAX model. The one line serial line controller is identified in CHARON-VAX with the name UART. The four serial lines controller is identified in CHARON-VAX with the name QUART.

UART is used in Qbus systems only (e.g. the MicroVAX/VAXserver 3600/3900).

QUART is used in SCSI (e.g. MicroVAX 3100 model 96/98, VAXstation 4000 model 90) and SCSI/Qbus systems (e.g. VAX4000 model 106/108). The last QUART line (*line[3]*) is the console port (known in VAX/VMS as *OPA0*).

CHARON-VAX console ports can be configured to connect to an external terminal via the host system COM/TTY port or can be connected via TCP/IP.

rts

Parameter	rts[<line>]
Type	Text string
Value	"On" - assert RTS (Request To Send) signal "Off" - clear RTS signal (default) "DTR" - assert RTS signal as soon as DTR signal is asserted (⚠ Applicable only for QUART)

dsr

Parameter	dsr[<line>]
Type	Text string
Value	"On" - always reports DSR signal asserted "Off" - always reports DSR signal deasserted "DSR" - use DSR signal of physical serial line (if configured) "CD", "DCD", "RLSD" - use CD (carrier detect) signal of physical serial line (if configured) This parameter is applicable only for line "2" of QUART. UART has no such parameter.

communication

Parameter	communication[<line>]
Type	Text string
Value	"ASCII" - for connection to terminals (default) "BINARY" - for serial lines carrying binary (packet) protocols, which are used mainly for communicating with PLCs

line

Parameter	line[<line>]
Type	Identifier
Value	This parameter is used to connect a particular serial line interface to the controller. See below for details.

Note that the "line" parameter in the table above is applicable only in the case of QUART.

All the values in this table are case insensitive.

Example:

```
set QUART rts[2]="DTR"
set QUART dsr[2]="On"
set QUART communication[2]="binary"
```

Line 2 of the QUART is the only one which can be used for connecting modems. Therefore, the "`DSR`" parameter for that line (i.e. "dsr[2]") is internally set to the appropriate value ("CD") but can be changed from the configuration file. Values for the "rts" and "dsr" parameters for the lines other than 2 are not visible for any applications running on CHARON-VAX / CHARON-PDP.

Serial line controllers

Asynchronous serial line multiplexers are capable of serving up to 8 asynchronous serial lines (the DHW42-BA supports 16 lines).The following asynchronous serial line multiplexers are supported:

VAX model	Asynchronous serial line emulation
MicroVAX II, MicroVAX 3600, MicroVAX 3900, VAXserver 3600, VAXserver 3900 (QBUS systems)	CXA16, CXB16, CXY08, DHQ11, DHV11, DZV11, DZQ11, DL11, DLV11, DZ11
MicroVAX 3100 - 96, MicroVAX 3100 - 98 (SCSI systems)	DHW42-AA, DHW42-BA, DHW42-CA
VAX4000 - 106, VAX4000 – 108, VAX4000 – 700, VAX4000 – 705 (QBUS/SCSI systems)	CXA16, CXB16, CXY08, DHQ11, DHV11, DZV11, DZQ11, DLV11, DHW42-AA, DHW42-BA, DHW42-CA
VAX6310, VAXstation 4090	N/A

The following names are used for the multiplexers:

Device name	Module name
DHV11	DHV11
DHQ11	DHV11
CXY08	DHV11
CXA16	DHV11
CXB16	DHV11
DHW42AA	DHV11
DHW42BA	DHV11
DHW42CA	DHV11
DZV11, DZ11	DZ11
DZQ11	DZ11
DL11, DLV11	DL11

The following example loads an instance of an asynchronous serial line multiplexer:

```
load DHQ11/DHV11 TXA
```

> Only one instance of DHW42 can be loaded. There is no restriction on the number of the other multiplexers.

The multiplexers offer the following configuration parameters, specified with the "set" command:

address

Parameter	address
Type	Numeric
Value	Specifies CSR address. The address must be valid QBUS 22-bit wide address in I/O space.Default values are 017760440 for DHV11-family controllers and 017760100 for DZV11/DZQ11, which are the factory settings for asynchronous serial line multiplexers. > This parameter is not applicable to DHW42-xx serial line controllers

vector

Parameter	vector
Type	Numeric
Value	Specifies interrupt vector.Default value is 0300 which is the factory setting for asynchronous serial line multiplexers. > This parameter is not applicable to DHW42-xx serial line controllers

line

Parameter	line[N] N=0…3 (7,15)
Type	Identifier
Value	Specifies a name of the serial line interface object in configuration to which the N-th line of the multiplexer is connected. See below for details.

communication

Parameter	communication[N] N=0…4 (7,15)
Type	Text String
Value	• "ASCII" - for connection to terminals (default) • "BINARY" - for serial lines carrying binary (packet) protocols, which are used mainly for communicating with PLCs

rts

Parameter	rts[N] N=0…3 (7,15)
Type	Text String
Value	Controls RTS signal of the Nth line of the multiplexer. • "On" - assert RTS (Request To Send) signal • "Off" - clear RTS signal (default) • "DTR" - assert RTS signal as soon as DTR signal is asserted When left blank (initial state), the level of the RTS signal is as requested by the VAX software.

dsr

Parameter	dsr[N] N=0…(7,15)
Type	Text String
Value	• "On" - always reports DSR signal asserted • "Off" - always reports DSR signal deasserted • "DSR" - use DSR signal of physical serial line (if configured) • "CD", "DCD", "RLSD" - use CD (carrier detect) signal of physical serial line (if configured) This parameter is applicable only for DZV11 and DZQ11 serial lines controllers

tx_q_max_depth

Parameter	tx_q_max_depth[N] N=0…3 (7,15)
Type	Numeric
Value	Specifies depth of the TX FIFO for the N-th line of the multiplexer. Possible values are 1…1000, initially it is set to 1 to properly represent the hardware limitation of certain multiplexers. Values greater than 1 improve transmission rate of corresponding line, but break correspondence to the original hardware. This parameter is applicable only for DHV11 serial lines controller

To load several instances of Qbus multiplexers, use the "address" and "vector" parameters. Both "address" and "vector" parameter values must be unique for every instance of a QBUS multiplexer. Read the VAX hardware documentation and the VM system management documentation to understand how to correctly assign the "address" and "vector" parameters.

Example of loading 2 instances of DHV11:

```
load DHV11/DHV11 TXA address=017760440 vector=0300
load DHV11/DHV11 TXB address=017760460 vector=0310
```

Example of loading DHW42CA:

```
load DHW42CA/DHV11 TXA
set TXA communication[0]="binary"
```

Mapping Serial line controllers to system resources

Types of serial line mapping

Type	Function
physical_serial_line	This type of mapping associates some TTY port on host system with an emulated VAX serial line controller virtual "line".The TTY port can be physical hardware port or a logical TTY port.
virtual_serial_line	This type of mapping associates a network connection on the host system with an emulated VAX serial line controller virtual "line"
operator_console	This type of mapping associates the current TTY console with the OPA0 console port (if CHARON-VAX does not run as service)

Example:

```
load physical_serial_line OPA0
```

physical_serial_line

line

Parameter	line
Type	Text string
Value	A defined TTY port on host system: "/dev/tty<N>" - virtual serial lines "/dev/ttyS<N>" - onboard serial lines "/dev/ttyUSB<N>" - modem or usb serial lines adapters "/dev/tty<XXX>" - proprietary (depending on a driver) devices such as DIGI or MOXA cards A specific account for running CHARON ("charon") does not allow usage of virtual consoles "/dev/tty<N>" as CHARON consoles. If you plan to map CHARON console or serial lines to "/dev/tty<N>" use only "root" account for CHARON running.

baud

Parameter	baud
Type	Numeric
Value	Forces the baud rate of the corresponding TTY port to a specified value.The variety of supported values depends on the underlying physical communication resource (TTY port). The most widely used values are: 300, 1200, 9600, 19200, 38400. **Example**: `set OPA0 baud=38400`

break_on

Parameter	break_on
Type	Text string
Value	Specifies what byte sequences received over the physical serial line will trigger a HALT command. ⚠ This parameter works only for the console line (for the one UART line and "line[3]" of QUART). Specify the following values: "Ctrl-P", "Break" or "none" ("none" disables triggering HALT condition). **Example**: `set OPA0 break_on="Ctrl-P"` The default value is "Break" for line 3 of QUART and "none" for other lines.

stop_on

Parameter	stop_on
Type	Text string
Value	Specifies what byte sequences received over the physical serial line will trigger a STOP condition. ⚠ This parameter works only for the console line (for the one UART line and "line[3]" of QUART). The STOP condition causes CHARON-VAX to exit.Specify the value as the following: "F6" or "none" ("none" disables triggering STOP condition). **Example**: `set OPA0 stop_on="F6"` The default value is "none". Setting "F6" triggers the STOP condition upon receipt of the "`<ESC>[17~`" sequence. Terminals usually send these sequences on pressing **F6** button

log

Parameter	log
Type	Text string

Value	A string specifying a file name to store content of console sessions or a directory where log files for each individual session will be stored.If an existing directory is specified, CHARON-VAX automatically enables creation of individual log files for each session.
	
	If the "log" parameter is omitted, CHARON-VAX does not create a console log.
	
	Examples:
	
	```set OPA0 log="log.txt"```
	
	```set OPA0 log="/opt/charon/logs"```

log_file_size

Parameter	log_file_size
Type	Text string

Value	If log rotation is enabled, the log_file_size parameter determines the log file size threshold at which the log is automatically rotated.
	
	• "unlimited" or "0" (default) - the feature is disabled
	• "default" - default size is used (4Mb)
	• <size>[KMG] - size of the current log file in bytes with additional multipliers:
	   • K - Kilobyte - multiply by 1024
	   • M - Megabyte - multiply by 1024*1024
	   • G - Gigabyte - multiply by 1024*1024*1024
	
	Examples:
	
	```set OPA0 log_file_size="default"```
	
	```set OPA0 log_file_size=10M```
	
	⚠ Minimum log file size is 64K, maximum is 1G. Setting size less then 64K effectively makes the log file unlimited.

Example of mapping a console line to an onboard serial line:

```
load physical_serial_line OPA0
set OPA0 line="/dev/ttyS1"
```

virtual_serial_line

host

Parameter	host
Type	Text string
Value	A remote host's IP address or a host name (and optional remote TCP/IP port number) for the virtual serial line connection. If omitted, the virtual serial line does not initiate a connection to the remote host and will listen for incoming connection requests. Specify the value in the following form: ```set OPA0 host="<host-name>[:<port-no>]"``` If "<port-no>" is not specified, the virtual serial line uses the TCP/IP port number specified by the "port" parameter (see below).

port

Parameter	port
Type	Numeric
Value	TCP/IP port number for the virtual serial line. A virtual serial line always listens on this port for incoming connection requests.

break_on

Parameter	break_on
Type	Text string
Value	Specifies what byte sequences received over virtual serial line must trigger HALT command. ⚠ This parameter works only for console line (for CHARON-VAX it is the only line of UART and the "line[3]" of QUART). Specify the following values: "Ctrl-P", "Break" or "none" to disable triggering HALT condition. **Example**: ```set OPA0 break_on="Ctrl-P"``` The default value is "Break" for line 3 of QUART and "none" for other lines.

stop_on

Parameter	stop_on
Type	Text string
Value	Specifies what byte sequences received over the virtual serial line will trigger a STOP condition. The STOP condition causes CHARON-VAX to exit. ⚠️ This parameter works only for console line (for CHARON-VAX it is the only line of UART and the "line[3]" of QUART). Specify the value as the following: "F6" or "none" ("none" disables triggering STOP condition). **Example**: `set OPA0 stop_on="F6"` The default value is "none".Setting "**F6**" triggers the STOP condition upon receipt of the "`<ESC>[17~`" sequence.

log

Parameter	log
Type	Text string

Value	A string specifying a file name to store content of console sessions or a directory where log files for each individual session will be stored.If an existing directory is specified, CHARON-VAX automatically enables creation of individual log file for each session.
	
	If the "log" parameter is omitted CHARON-VAX does not create any console log.
	
	Examples:
	
	`set OPA0 log="log.txt"`
	
	`set OPA0 log="/opt/charon/logs"`

log_file_size

Parameter	log_file_size
Type	Text string

Value	If log rotation is enabled, the log_file_size parameter determines the log file size threshold at which the log is automatically rotated.
	
	• "unlimited" or "0" (default) - the feature is disabled
	• "default" - default size is used (4Mb)
	• <size>[KMG] - size of the current log file in bytes with additional multipliers:
	  • K - Kilobyte - multiply by 1024
	  • M - Megabyte - multiply by 1024*1024
	  • G - Gigabyte - multiply by 1024*1024*1024
	
	Examples:
	
	`set OPA0 log_file_size="default"`
	
	`set OPA0 log_file_size=10M`
	
	⚠️ Minimum log file size is 64K, maximum is 1G. Setting size less then 64K effectively makes the log file unlimited.

Example of mapping a console line to an onboard serial line:

```
load virtual_serial_line OPA0
set OPA0 port=10003 stop_on="F6"
```

Notes on "virtual_serial_line" options:

1. Use the combination of "port" and "host" parameters as follows to connect a 3rd party terminal emulator or similar program.

```
load virtual_serial_line/chserial TTA0 host="192.168.1.1" port=10000
```

In this example CHARON-VAX connects to port 10000 of a host with TCP/IP address "192.168.1.1" and at the same time it accepts connections on local port 10000.

2. It is possible to specify a port on a remote host (note that CHARON always acts as a server). The syntax is:

```
load virtual_serial_line/chserial TTA0 host="192.168.1.1:20000" port=10000
```

In this example CHARON-VAX accepts connection on local port 10000 and connects to remote port 20000 of a host with TCP/IP address "192.168.1.1"

Note that the examples above are mainly used for inter-CHARON communications. They are used to connect CHARON-VAX to an application that communicates to CHARON-VAX as described below.
Example of two CHARON systems connected to each other:*On host "A":*

```
load virtual_serial_line/chserial TXA0 port=5500 host="B"
```

On host "B":

```
load virtual_serial_line/chserial TXA0 port=5500 host="A"
```

On two hosts executing CHARON-VAX, the two TXA0 lines connect to each other, thus creating a "serial" cable between the two emulated VAXes. The sequential order in which the instances of CHARON-VAX are started makes no difference.

operator_console

break_on, stop_on

Parameter	break_on, stop_on
Type	Text string

Value	These two parameters are hardcoded to the following values and cannot be changed:
	
	```stop_on="F6"```
	```break_on="Ctrl-P,F5"```

Example:

```
load operator_console OPA0
```

"ttyY" notation specifics

Note that the "$ttyY$" notation can have different forms depending on the nature of the device used:

1. Linux virtual tty (switchable by **alt+F1** - **atl+F12** on a text console) – are represented as "/dev/tty<N>" where N is from 0 to 11. Those tty devices must be free from the Linux "getty/mgetty" and similar programs (specified in "/etc/inittab")
2. Onboard serial lines are represented as "/dev/ttyS<N>" where N is a number. For example "/dev/ttyS1"
3. Proprietary (depending on a driver) devices are represented as "/dev/tty<XXX>" where XXX is a complex letter/number notation. For example "/dev/ttyR01" is the first port of a MOXA card and "/dev/ttyaa" stands for the first port of a DIGI card.

Linking serial controller port to host connection

The final step of CHARON-VAX serial line configuration is the association of each loaded serial port with a CHARON-VAX host connection instance as follows:

```
set <serial controller instance name> line[<line number>]=<serial line instance name>
```

Example:

```
set quart line[0]=TTA0
```

This command connects the first serial line ("line[0]") of a "QUART" serial line controller to a CHARON-VAX connection instance named "TTA0". As explained earlier, TTA0 may be a virtual serial line connected to port, or a physical serial line connected to host serial port or virtual terminal. In an example below, the command connects the sixth serial line of a previously loaded controller (named "TXA") to "TTA1". "TTA1" could be defined, for example, as a physical serial line connected to COM/TTY port:

```
set TXA line[5]=TTA1
```

Disks and tapes

Contents

- MSCP and TMSCP Controllers
- SCSI Controllers
- DSSI Subsystem
- CI Subsystem
- Finding the target "/dev/sg" device

MSCP and TMSCP Controllers

Table of Contents

Introduction

CHARON-VAX provides MSCP controllers for hardware disks (including floppy and CD/DVD) and disks images. TMSCP controllers provide support for hardware tapes and tape images.

MSCP and TMSCP controllers are added to the configuration using the "load" command. The individual units are defined by using the container parameter.

> MSCP devices appear in VMS as DUA for the first controller and DUB for the second controller, etc.
> TMSCP devices appear in VMS as MUA, MUB, etc.

When adding multiple MSCP or TMSCP controllers, follow Qbus addressing conventions.

When a tape or disk image connected to an emulated TMSCP or MSCP controller is dismounted in VAX/VMS, it is disconnected from CHARON-VAX and can be manipulated. It may even be replaced with a different disk image if it keeps the same name. This capability may be useful when designing back-up and restore procedures. When copying CHARON-VAX disk images while CHARON-VAX is running, please take care to minimize the risk of overloading a heavily loaded CHARON-VAX host system. For example using a sequential series of simple ftp binary copies is less resource intensive and thus less disruptive than multiple, simultaneous copies.

Empty disk images are created with the "mkdskcmd" utility. Tape images (vtapes) will be created if they don't exist.

CHARON-VAX is able to boot disk images of any VAX/VMS version (for VAX/VMS starting with 4.5 or higher for MicroVAX II or VAX 3600 and VAX/VMS 5.5-2 or higher for the VAX4000).

RQDX3 Controller

The CHARON-VAX QBUS system provides support for RQDX3 disk controllers. The original RQDX3 disk controller is capable of serving up to 4 disk units. CHARON-VAX extends this limit so that the RQDX3 disk controller can be configured with up to 256 disk units. Normally all 256 disks can be connected to one MSCP disk controller but, if an application does intensive simultaneous I/O to more than 16 disks on one MSCP controller, it is recommended to configure additional RQDX3 controllers.

Use the following command to load an instance of an RQDX3 disk controller:

```
load RQDX3/RQDX3 <logical name>
```

Example:

```
load RQDX3/RQDX3 DUA
```

The RQDX3 offers the following configuration parameters, which can be specified with the "set" command:

address

Parameter	address
Type	Numeric
Value	Specifies the CSR address. The address must be a valid QBUS 22-bit wide address in IO space. Initial value is 017772150 which is the factory setting for the RQDX3 disk controller. Use the "address" parameter for loading several instances of RQDX3. The "address" parameter value must be unique for every instance of the controller.

max_n_of_units

Parameter	max_n_of_units
Type	Numeric
Value	Specifies the maximum number of units supported by the controller. Possible values are 4…9999. Default is 9999.

container

Parameter	container[N]
	N=0…9999
Type	Text String

Value	Specifies the location of the disk container. It can be either the name of a ".vdisk" file or the name of a physical disk:
	
	■ **Local fixed disks (IDE, SCSI, SATA)**
	   ■ "/dev/sd<L>" where "L" is letter
	   ■ "/dev/disk/by-id/..." - addressing by the disk ID, for example "/dev/disk/by-id/ata-ST1000DM003-9YN162_S1D01QJ4"
	   ■ "/dev/disk/by-label/..." - addressing by the disk label, for example "/dev/disk/by-label/MyStorage"
	   ■ "/dev/disk/by-uuid/..." - addressing by the disk UUID, for example "/dev/disk/by-uuid/0e808a2f-cdd3-4944-a245-f729ffd73882"
	
	> Since "/dev/sd<L>" addressing is not persistent, so it is strongly recommended to use "/dev/disk/by-id/wwn-*." syntax instead to refer the disk by its WWID - especially in the enviroments utilizing FC and SAN storages.
	
	ℹ️ It is also possible to use not a whole disk, but previously created partitions on it. In this case the syntax is the following: "/dev/sd<L><N>" where N is the number of partition to be used.
	
	■ **Floppy drives**
	   ■ "/dev/fd<N>"
	
	■ **CD-ROM and DVD drives (IDE, SCSI, …)**
	   ■ "/dev/cdrom<N>",
	   ■ "/dev/sr<N>"
	
	■ **Multipath disks**
	   ■ "/dev/dm-<N>",
	   ■ "/dev/mapper/mpath<N>",
	   ■ "/dev/mapper/disk<N>"

media_type

Parameter	media_type[N]
	N=0…9999
Type	Text String
Value	Overrides default (automatically determined) MSCP media type of the device. Syntax: ``` "<device-name>[,<device-type>]" ``` where: • <device-name> is one of "DU", "DK", "SCSI", "DI", "DSSI", "DJ" • <device-type> is of the form "LLD" or "LLLD", where "L" is letter from A through Z, and "D" is decimal number from 0 through 99 If not specified, the device name is set to "DU", and the device type is selected based on disk size. Initialy not specified.

geometry

Parameter	geometry[N]
	N=0…9999
Type	Text String
Value	This formatted string value specifies the explicit geometry of the disk storage element with DSSI node id N and MSCP unit number N. This parameter is not applicable to tape storage elements.
	The string format is <X>"/"<Y>["/"<Z>] where:
	1. X is number of sectors on track;
	2. Y is number of tracks on cylinder;
	3. Z (optional) is the number of cylinders on the unit. If omitted, Z is calculated based on X, Y and the total number of sectors on the unit that reflects the size of the disk storage element;
	If this parameter is not set, CHARON-VAX will configure the geometry based on the most probable disk type.

use_io_file_buffering

Parameter	use_io_file_buffering[N]
	N=0…9999
Type	Boolean
Value	Enables use of host OS I/O buffering.
	Initially set to "false" (buffering disabled).

Example 1:

```
load RQDX3/RQDX3 DUA address=017772150 max_n_of_units=4

set DUA container[0] = "/charon/disks/rx23.vdisk"
set DUA container[1] = "/dev/sdb"

load RQDX3/RQDX3 DUB address=017760334

set DUB container[5] = "/dev/cdrom"
```

In the above example, "rx23.vdisk" will be seen in VMS as DUA0, "/dev/sdb" as DUA1 and "/dev/cdrom" as DUB5.

Example 2:

```
load RQDX3/RQDX3 DIA address=017772150 max_n_of_units=4

set DIA container[0] = "/charon/disks/rx23.vdisk"
set DIA media_type[0] = "dssi"
set DIA container[1] = "/dev/sdc2"
set DIA media_type[1] = "dssi"
```

In the above example, "rx23.vdisk" will be seen in VMS as DIA0 and "/dev/sdc2" as DIA1.

TQK50 and TUK50 Controllers

The CHARON-VAX QBUS system provides support for the TQK50 tape controller. UNIBUS systems support the TUK50 tape controller.

The original TQK50/TUK50 tape controller is capable of serving only 1 tape unit. CHARON-VAX extends the limit to 10000 tape units.

Use the following commands to load an instance of a TQK50/TUK50 tape controller:

```
load TQK50/TQK50 <logical name 1>
load TUK50/TUK50 <logical name 2>
```

Example:

```
load TQK50/TQK50 MUA1
load TUK50/TUK50 MUA2
```

The TQK50/TUK50 controllers have the following configuration parameters, which can be specified with the "set" command:

address

Parameter	address
Type	Numeric
Value	Specifies the CSR address. The address must be a valid QBUS 22-bit wide address in IO space for TQK50 and a valid QBUS 18-bit wide address in I/O space for TUK50. The initial values are 017774500 (TQK50) and 0774500 (TUK50) which is the factory setting for those tape controllers. Use the "address" parameter to load several instances of TQK50/TUK50. The "address" parameter value must be unique for each instance of TQK50/TUK50.

container

Parameter	container[N] N=0…9999
Type	Text String
Value	Specifies the location of the tape container. It can be either the name of a ".vtape" (".mtd") file or the name of a physical tape drive: "dev/sg<N>"– for the local physical tape drive.

media_type

Parameter	media_type[N] N=0…9999
Type	Text String
Value	Overrides default (automatically determined) TMSCP media type of the device. Syntax: `"<device-name>[,<device-type>]"` where: • <device-name> is one of "MU", "MK", "SCSI", "MI", "DSSI", or "MJ" • <device-type> is of the form "LLD" or "LLLD", where "L" is letter from A through Z, and "D" is decimal number from 0 through 99 If not specified, the device name is set to "MU", and the device type is set to "TK50" Initially not specified.

geometry

Parameter	geometry[N] N=0.9999
Type	Text String

Value	Specifies the size of the tape image and (optionally) the size of an "early-warning" area at the end of tape image.
	
	Syntax:
	
	`"<image-size>[, <early-warning-zone-size>]"`
	
	where:
	
	• \<image-size> is the tape size in MB
	• \<early-warning-zone-size> is the size (in KB) of space left on the tape that, when reached, issues a warning to the OS. If omitted, 64K is assumed.
	
	Example:
	
	`load TQK50 MUA`
	`set MUA geometry[0] = 90`

Example:

```
load TQK50 MUA address=017774500

set MUA container[0] = "/dev/sg4"
set MUA container[1] = "/charon/tapes/tape1.vtape"
```

Multi-volume tape images may be handled as follows:

```
set MUA container[0] = "..."
set MUA container[1] = "..."
set MUA container[2] = "..."
set MUA container[3] = "..."
```

Once this configuration is established, the following VMS command (for example) could be used:

```
$ BACKUP MUA0:BACKUP.SAV,MUA1,MUA2,MUA3/SAVE_SET DUA0:...
```

KDM70 Controller

KDM70 is an MSCP/TMSCP disk and tape controller for the VAX 6000.

The CHARON-VAX virtual KDM70 controller supports 9999 disks and tapes instead of the 8 disk limitation of the original hardware. This design modification has the advantage of using only one XMI slot for up to 9999 disk and tape devices.

The I/O behavior of the virtual KDM70 is as follows:
• Up to 16 connected disks operate in parallel without any I/O performance degradation.
• For systems with more than 16 heavily used disks, configure two controllers and distribute the heavily loaded disks evenly.
• As in the hardware KDM70, VMS can be booted only from the first 10 devices on the KDM70 (DU0 - DU9).
• Hardware KDM70's do not support tape drives. The virtual KDM70's support a transparent extension for data tapes (boot from tape is not supported).

The line below loads an emulated KDM70 storage controller:

```
load KDM70/KDM70 PUA
```

The KDM70 emulation has the following configuration parameters:

xmi_node_id

Parameter	xmi_node_id
Type	Numeric
Value	Specifies the XMI slot in which virtual KDM70 controller is placed. For CHARON-VAX/66X0 a free slot between 10 (A) and 14 (E) must be chosen.

container

Parameter	container[N] N=0…9999
Type	Text String
Value	Possible values of the parameter are strings in one of the following forms: **Physical disk drives** ■ "/dev/sd<L>", where L is letter ⓘ It is also possible to use not a whole disk, but previously created partitions on it. In this case the syntax is the following: "/dev/sd<L><N>" where N is the number of partition to be used. ■ "/dev/disk/by-id/..." - addressing by the disk ID, for example "/dev/disk/by-id/ata-ST1000DM003-9YN162_S1D01QJ4" ■ "/dev/disk/by-label/..." - addressing by the disk label, for example "/dev/disk/by-label/MyStorage" ■ "/dev/disk/by-uuid/..." - addressing by the disk UUID, for example "/dev/disk/by-uuid/0e808a2f-cdd3-4944-a245-f729ffd73882" Since "/dev/sd<L>" addressing is not persistent, so it is strongly recommended to use "/dev/disk/by-id/wwn-*" syntax instead to refer the disk by its WWID - especially in the enviroments utilizing FC and SAN storages. **Physical tape drives** ■ "/dev/sg<X>", where X is 0, 1, … **Floppy drives** ■ "/dev/fd<X>", where X is 0, 1, … **CD-ROM drives (read-only)** ■ "/dev/cdrom<X>" or ■ ("/dev/sr<X>"), where X is 0, 1, … **Multipath disks** ■ "/dev/dm-<N>", ■ "/dev/mapper/mpath<N>", ■ "/dev/mapper/disk<N>" **CHARON-VAX disk images** ■ [<path-name>"/"]<file-name>[".vdisk"] **CHARON-VAX tape images** ■ [<path-name>"/"]<file-name>".vtape" This parameter is initially not set, thus creating NO storage elements on the controller

media_type

Parameter	media_type[N] N=0…9999
Type	Text String
Value	Overrides default (automatically determined) MSCP media type of the device. Syntax: ```"<device-name>,<device-type>"``` where: • <device-name> is one of "DU", "MU", "DK", "MK", "SCSI", "DI", "MI", "DSSI", "DJ", "MJ" • <device-type> is of the form "LLD" or "LLLD", where "L" is letter from A through Z, and "D" is decimal number from 0 through 99 If not specified, the device name is set to "DI", and the device type is selected based on disk size for disk storage elements. For tape storage elements, the device name and type are set to "MI" and "TF86", respectively. Initialy not specified.

geometry

Parameter	geometry[N] N=0…9999
Type	Text String
Value	This formatted string value specifies the explicit geometry of the disk storage element with DSSI node id N and MSCP unit number N. This parameter is not applicable to tape storage elements. The string format is <X>"/"<Y>["/"<Z>] where: 1. X is number of sectors on track 2. Y is number of tracks on cylinder 3. Z (optional) is the number of cylinders on the unit. If omitted, Z is calculated based on X, Y and the total number of sectors on the unit that reflects the size of the disk storage element If this parameter is not set, CHARON-VAX will configure the geometry based on the most probable disk type. Initially not set. The syntax described above is applicable only to disk storage elements. If the container is a tape image, the following format is used instead: Syntax: ```"<image-size>[, <early-warning-zone-size>]"``` where: • <image-size> is the tape size in MB • <early-warning-zone-size> is a size (in KB) of a space left on the tape when a warning to the OS is issued. If omitted, 64K is assumed.

use_io_file_buffering

Parameter	use_io_file_buffering[N] N=0…9999
Type	Boolean
Value	Enables use of host OS I/O buffering. Initially set to "false" (buffering disabled).

Example:

Create a KDM70 (T)MSCP controller in XMI slot 10:

```
load KDM70/KDM70 PUA xmi_node_id=10
```

Configure on this controller a system disk to show up as DUA0: in VMS:

```
set PUA container[0]="/charon/disks/vms72-66X0.vdisk"
```

Configure a user disk to show up as DUA1: in VMS:

```
set PUA container[1]="/charon/disks/usertest.vdisk"
```

Configure the first SCSI tape drive connected to the host to show up as MUA4: in VMS:

```
set PUA container[4]="/dev/sg0"
```

The file my_tape.vtape in the default directory is used by VMS as MUA5:

```
set PUA container[5]="/charon/tapes/my_tape.vtape"
```

The first host system CD-ROM can be used to read VMS CDs and shows up as DUA9:

```
set PUA container[9]="/dev/sr0"
```

The host system floppy drive "/dev/fd0' can be used inVMS as DUA10:

```
set PUA container[10]="/dev/fd0"
```

> The virtual KDM70 examines the file extension (`vdisk or vtape`) to distinguish between a disk image and a tape image, Configured physical devices or tape/disk images that do not exist on the host system will, in general, cause VAX/VMS to report the unit offline. In some cases this will result in a VMS BUG CHECK. In this case, an error message will be written to the log file.

KDB50 Controller

KDB50 is an MSCP controller for the VAX 6000. The CHARON-VAX virtual KDB50 controller supports up to 9999 disks instead of the 4 disk limitation of the original hardware. This design modification has the advantage of using only one VAXBI slot for up to 9999 disk and tape devices.

The I/O behavior of the virtual KDB50 is as follows:

1. Up to 16 connected disks operate in parallel without any I/O performance degradation.
2. For systems with more than 16 heavily used disks, configure two controllers and distribute the heavily loaded disks evenly.
3. Like the hardware KDB50, VMS can boot only from the first 10 devices on the KDB50 (DU0 - DU9).

The line below loads an emulated KDB50 storage controller:

```
load KDB50 PUA
```

The KDB50 emulation has the following configuration parameters:

vax_bi_node_id

Parameter	vax_bi_node_id
Type	Numeric
Value	Specifies the VAXBI slot in which the virtual KDB50 controller is placed. For CHARON-VAX a free slot between 1 (1) and 15 (F) must be chosen.Initially set to 14.

container

Parameter	container[N] N=0…9999
Type	Text String
Value	Possible values of the parameter are strings in one of the following forms: ■ **Physical disk drives** 　■ "/dev/sd<L>", where L is letter 　■ "/dev/disk/by-id/..." - addressing by the disk ID, for example "/dev/disk/by-id/ata-ST1000DM003-9YN162_S1D01QJ4" 　■ "/dev/disk/by-label/..." - addressing by the disk label, for example "/dev/disk/by-label/MyStorage" 　■ "/dev/disk/by-uuid/..." - addressing by the disk UUID, for example "/dev/disk/by-uuid/0e808a2f-cdd3-4944-a245-f729ffd73882" 　　Since "/dev/sd<L>" addressing is not persistent, so it is strongly recommended to use "/dev/disk/by-id/wwn-*." syntax instead to refer the disk by its WWID - especially in the enviroments utilizing FC and SAN storages. 　　ⓘ It is also possible to use not a whole disk, but previously created partitions on it. In this case the syntax is the following: "/dev/sd<L><N>" where N is the number of partition to be used. ■ **Floppy drives** 　■ "/dev/fd<X>", where X is 0, 1, … ■ **CD-ROM drives (read-only)** 　■ "/dev/cdrom<X>" or 　■ ("/dev/sr<X>"), where X is 0, 1, … ■ **Multipath disks** 　■ "/dev/dm-<N>", 　■ "/dev/mapper/mpath<N>", 　■ "/dev/mapper/disk<N>" ■ **CHARON-VAX disk images** 　■ [<path-name>"/"]<file-name>[".vdisk"] This parameter is initially not set, thus creating NO storage elements on the controller

media_type

Parameter	media_type[N] N=0…9999
Type	Text String
Value	Overrides default (automatically determined) MSCP media type of the device. Syntax: ```"<device-name>,<device-type>"``` where: • <device-name> is one of "DU", "DK", "SCSI", "DI", "DSSI", "DJ" • <device-type> is of the form "LLD" or "LLLD", where "L" is letter from A through Z, and "D" is decimal number from 0 through 99 If not specified, the device name is set to "DU", and the device type is selected based on disk size for disk storage elements. For tape storage elements, the device name and type are set to "MI" and "TF86", respectively.Initialy not specified.

geometry

Parameter	geometry[N] N=0…9999
Type	Text String
Value	This formatted string value specifies the explicit geometry of the disk storage element with MSCP unit number N. This parameter is not applicable to tape storage elements.T he string format is <X>"/"<Y>["/"<Z>] where: 1. X is number of sectors on track 2. Y is number of tracks on cylinder 3. Z (optional) is the number of cylinders on the unit. If omitted, Z is calculated based on X, Y and the total number of sectors on the unit that reflects the size of the disk storage element If this parameter is not set, CHARON-VAX will configure the geometry based on the most probable disk type.Initially not set.

use_io_file_buffering

Parameter	use_io_file_buffering[N] N=0…9999
Type	Boolean
Value	Enables use of host OS I/O buffering. Initially set to "false" (buffering disabled).

Example:

Create a KDB50 MSCP controller in VAX BI slot 1:

```
load KDB50/KDB50 PUA vax_bi_node_id=1
```

Configure on this controller a system disk to show up as DUA0: in VMS:

```
set PUA container[0]="/charon/disks/vms72-66X0.vdisk"
```

Configure a user disk to show up as DUA1: in VMS:

```
set PUA container[1]="/charon/disks/usertest.vdisk"
```

The first host system CD-ROM can be used to read VMS CDs and shows up as DUA9:

```
set PUA container[9]="/dev/sr0"
```

The host system floppy drive `/dev/fd0` can be used in VMS as DUA10:

```
set PUA container[10]="/dev/fd0"
```

> Configured physical devices or tape/disk images that do not exist on the host system will, in general, cause VAX/VMS to report the unit offline. In some cases this will result in a VMS BUG CHECK. In this case, an error message will be written to the log file.

Back to Table of Contents

SCSI Controllers

Table of Contents

Introduction

CHARON-VAX provides two SCSI controllers for the SCSI and SCSI/QBUS models of VAX.

Hardware disks, disk images, hardware tapes, tape images, floppy devices and CD-ROM devices can be connected to these SCSI controllers. Each device has to be configured to connect to a specific SCSI address in CHARON-VAX.

Use the following emulated device types to map real peripherals to the emulated SCSI devices:

Type of mapping	Description
virtual_scsi_disk	For disk image containers and physical disks
virtual_scsi_tape	For tape image containers
virtual_scsi_cdrom	For host CD-ROM and *.iso images
physical_scsi_device	For physical tapes and other SCSI devices connected to a host

CHARON-VAX disk/tape devices can be SCSI disk/tape devices connected to a host system or disk/tape containers that are presented to the operating system environment as files.

Two SCSI controllers are provided ("PKA" and "PKB") in CHARON-VAX, with 7 addresses each.

Beyond the capabilities of the hardware, VAX 3100/9x and 4000/10x, CHARON-VAX/XX implements extended SCSI addressing. Each of the seven device addresses, of a SCSI controller, supports up to eight disk/tape images. Thus the number of disks supported becomes 2x Controllers*7 addresses*8 Disks/Tapes, a total of 112 disks/tapes.

SCSI devices with the same ID but different LUNs (logical units) appear in the VAX console with different names. The naming convention is as follows:

Each SCSI device has the name in the form of "$xKct0n$:", where:

- "x" stands for the device type (D means disks, M means magnetic tapes, G is reserved by VAX/VMS for special purposes)
- "c" stands for the controller letter (A - the first controller, B - the second controller, ...)
- "t" stands for the SCSI device ID (usually 0 through 6, and 7 is allocated by the controller itself)
- "n" stands for a particular logical unit number, LUN.

Most of the 'normal' SCSI devices have only one logical unit - 0. Therefore, under normal conditions, disks in VAX/VMS appear as DKA0 (which is really DKA000), DKA100, DKA200, ..., tapes as MKA0 (which is really MKA000), MKA100, MKA200, ...

As soon as there is a disk/tape device with LUNs 0 and 1, VMS identifies them as, for example, DKA300 and DKA301 (MKA300 and MKA301) respectively.

The boot ROM of CHARON-VAX detects SCSI devices with multiple LUNs and builds proper device names for them.

ℹ To display a list of devices on the VAX console (SRM), enter ">>> `show device`".

This list is passed to VAX/VMS at boot time.

⚠ VAX/VMS creates devices only for logical unit 0 for each device detected in the boot ROM. To add additional logical units, use the following SYSGEN command:

```
$ MCR SYSGEN CONNECT DKxxx/NOADAPTER
```

where DKxxx (or MKxxx) stands for the correct VAX/VMS name of the logical unit to be connected. You can find its name from the SRM console using the ">>> `show scsi`" command.

This command is not boot persistent, so it must be included in the VAX/VMS "SYSTARTUP_VMS.COM" file to ensure it is executed with each startup.

Also note: that the following rules are applied for logical units.

1. Each SCSI device must implement logical unit 0.
2. A SCSI device must implement all logical unit numbers between the highest and the lowest numbers implemented.

Empty disk images can be created with the "mkdskcmd utility".

CHARON-VAX is able to boot from disk images of any VAX/VMS version, starting with 4.5 or higher for MicroVAX II or VAX 3600 and VMS 5.5-2 or higher for the VAX4000.

Mapping to host resources

Load a mapping device with the "load" command. Specify the name of the device instance, the emulated SCSI bus to connect the device to and the SCSI identifier of the CHARON-VAX device.

scsi_bus

Parameter	scsi_bus
Type	Identifier
Value	Name of the emulated SCSI disk controller: "pka" or "pkb"

scsi_id

Parameter	scsi_id
Type	Numeric
Value	A value between 0 and 7. This is the ID number of the emulated SCSI device. The SCSI adapter is preloaded with address 7. If required, set it to another value in the range of 0-7 from the VAX console.

> There is no direct correspondence between the host hardware SCSI ID and these CHARON-VAX SCSI addresses. Set the correspondence between the physical SCSI addresses on the host system and the CHARON-VAX SCSI bus ID in the configuration file.

Syntax:

```
load <instance type>/<module name> <instance name> scsi_bus=<bus name> scsi_id=<number>
```

Example:

```
load virtual_scsi_disk/chscsi pka_0 scsi_bus=pka scsi_id=0
```

CHARON-VAX/*XX* has only one preloaded SCSI adapter, named: PKA. If a second adapter (PKB) is required then add the following line to the configuration file before loading and configuring any device on the second SCSI adapter PKB:

```
include kzdda.icfg
```

"kzdda.icfg" loads the second SCSI adapter.

ⓘ OpenVMS version 5.5-2H4, or above, is required to use the "pkb" controller.

virtual_scsi_disk

Use the "virtual_scsi_disk" mapping for disk containers and physical disks. This is the most convenient way of connecting disks to SCSI adapters in CHARON-VAX

The "virtual_scsi_disk" mapping has the following parameters:

container

Parameter	container[N]
	N=0…7
Type	Text String
Value	A string containing the device name to map to the emulator.

Value (continued):

- **Disk images**
  - [<path-name>"/"]<file-name>[".vdisk"]

    Format: A string containing the full path to a disk container.

    If only the name of the disk container is specified, CHARON-VAX will look for the container in the folder CHARON is running from.

- **Local fixed disks (IDE, SCSI, SATA)**
  - "/dev/sd<L>" where "L" is letter
  - "/dev/disk/by-id/..." - addressing by the disk ID, for example "/dev/disk/by-id/ata-ST1000DM003-9YN162_S1D01QJ4"
  - "/dev/disk/by-label/..." - addressing by the disk label, for example "/dev/disk/by-label/MyStorage"
  - "/dev/disk/by-uuid/..." - addressing by the disk UUID, for example "/dev/disk/by-uuid/0e808a2f-cdd3-4944-a245-f729ffd73882"

    > Since "/dev/sd<L>" addressing is not persistent, so it is strongly recommended to use "/dev/disk/by-id/wwn-*" syntax instead to refer the disk by its WWID - especially in the enviroments utilizing FC and SAN storages.

    ⓘ It is also possible to use not a whole disk, but previously created partitions on it. In this case the syntax is the following: "/dev/sd<L><N>" where N is the number of partition to be used.

- **Multipath disks**
  - "/dev/dm-<X>",
  - "/dev/mapper/mpath<X>",
  - "/dev/mapper/disk<X>" where X is 0,1,2...

- **Floppy drives**
  - "/dev/fd<X>" where X is 0,1,2...

This parameter is initially not set, thus creating NO storage elements on the controller

media_type

Parameter	media_type[N] N=0…7
Type	Text String
Value	Overrides the PRODUCT ID in the default SCSI INQUIRY data. Valid values may contain uppercase letters, integers and spaces. The length of the string cannot exceed 16 characters. If not specified, synthetic SCSI INQUIRY data is returned containing a PRODUCT ID selected based on the disk size. Initially left unspecified.

geometry

Parameter	geometry[N] N=0…7
Type	Text String
Value	This formatted string value specifies the explicit geometry of the disk storage element The string format is <X>"/"<Y>["/"<Z>] where: "X" is the number of sectors per track; "Y" is the number of tracks per cylinder; "Z" (optional) is the number of cylinders on the unit. If omitted, Z is calculated based on X, Y and the total number of sectors on the unit that reflects the size of the disk storage element; If this parameter is not set, CHARON-VAX will configure the geometry based on the most probable disk type.

use_io_file_buffering

Parameter	use_io_file_buffering[N] N=0…7
Type	Boolean
Value	Enables use of the host OS I/O buffering. Initially set to "false" (buffering disabled).

removable

Parameter	removable[N] N=0…7
Type	Boolean
Value	Enables the logical unit to appear as a removable SCSI disk drive. Initially set to "false" (fixed, non-removable).

⚠ In the table above N stands for logical unit number. The first unit must be 0 with no gaps in subsequent numbering.

Example:

```
load virtual_scsi_disk/chscsi pka_0 scsi_bus=pka scsi_id=0

set pka_0 container[0] = "/charon/disks/disk1.vdisk"
set pka_0 container[1] = "/dev/sdc"
```

If only one LUN is configured, the LUN number can be omitted:

```
set pka_0 container = "/charon/disks/disk1.vdisk"
set pka_0 media_type = "RZ1ED"
```

When a virtual SCSI disk image is dismounted in VMS, it is no longer open by CHARON and may be copied. This capability can be useful when designing back-up and restore procedures. If copying CHARON-VAX disk images while CHARON-VAX is running, take care to minimize the risk of overloading the host system.

Unlike MSCP controlled disk images, a disk image connected to a SCSI controller as a virtual SCSI disk CANNOT be replaced by another disk image unless "removable" parameter is set for this particular disk image.

Example:

```
set pka_0 container = "/charon/disks/my_removable_disk.vdisk"
set pka_0 removable = true
```

Back to Table of Contents

virtual_scsi_tape

Use "virtual_scsi_tape" for tape containers. This is the most convenient way of connecting tapes to SCSI adapters in CHARON-VAX.

"virtual_scsi_tape" has the following parameters:

container

Parameter	container[N] N=0…7
Type	Text String
Value	A string containing the full path to a tape container. If the specified tape image does not exist, CHARON-VAX creates it. If only the name of the tape container is specified, CHARON-VAX will look for the container in the folder CHARON is running from.

media_type

Parameter	media_type[N] N=0…7
Type	Text String
Value	Overrides PRODUCT ID in the default SCSI INQUIRY data. Valid values may contain uppercase letters, integers and spaces. Length of this string cannot exceed 16 characters. By default the PRODUCT ID returned is "TLZ04"

geometry

Parameter	geometry[N] N=0…7
Type	Text String

Value	Specifies the size of the tape image and (optionally) the size of the "early-warning" area at the end of the tape image.
	
	Syntax:
	
	```"<image-size>[, <early-warning-zone-size>]"```
	
	where:
	
	• <image-size> is the tape size in MB
	• <early-warning-zone-size> is the size (in KB) of the space remaining on the tape when a warning to the OS is issued. If omitted, 64K is assumed.
	
	Example:
	
	```load virtual_scsi_tape/chscsi pka_0```
	```set pka_0 geometry[0] = 90```

⚠ In the table above N stands for logical unit number. The first unit must be 0 with no gaps in subsequent numbering.

Example:

```
load virtual_scsi_tape/chscsi pka_0 scsi_bus=pka scsi_id=0

set pka_0 container[0] = "/charon/tapes/tape1.vtape"
set pka_0 container[1] = "/charon/tapes/tape2.vtape"
```

If only one LUN is configured, the LUN number can be omitted:

```
set pka_0 container = "/charon/tapes/tape1.vtape"
set pka_0 media_type = "TLZ08"
```

[Back to Table of Contents](#)

virtual_scsi_cdrom

Use "virtual_scsi_cdrom" to connect IDE or SATA host CD-ROM or *.iso file to CHARON-VAX.

"virtual_scsi_cdrom" has the following parameters:

container

Parameter	container[N] N=0…7
Type	Text String
Value	A string containing the device name to map to the emulator. • **ISO images** • [<path-name>"/"]<file-name>[".iso"] Format: A string containing the full path to an *.iso image. If only the name of the *.iso image is specified, CHARON-VAX will look for it in the folder CHARON is running from. • **CD-ROM or DVD drives** • "/dev/cdrom<X>" or • "/dev/sr<X>" where X is 0,1,2... This parameter is initially not set, thus creating NO storage elements on the controller

media_type

Parameter	media_type[N] N=0…7
Type	Text String
Value	Overrides the PRODUCT ID in the default SCSI INQUIRY data. Valid values may contain uppercase letters, integers and spaces. The length of the string cannot exceed 16 characters. If not specified, synthetic SCSI INQUIRY data is returned containing a PRODUCT ID selected based on the disk size. Initially left unspecified.

geometry

Parameter	geometry[N] N=0…7
Type	Text String
Value	This formatted string value specifies the explicit geometry of the disk storage element The string format is <X>"/"<Y>["/"<Z>] where: • "X" is the number of sectors per track; • "Y" is the number of tracks per cylinder; • "Z" (optional) is the number of cylinders on the unit. If omitted, Z is calculated based on X, Y and the total number of sectors on the unit that reflects the size of the disk storage element; If this parameter is not set, CHARON-VAX will configure the geometry based on the most probable disk type.

use_io_file_buffering

Parameter	use_io_file_buffering[N] N=0…7
Type	boolean
Value	Enables use of the host OS I/O buffering. Initially set to "false" (buffering disabled).

⚠️ In the table above N stands for logical unit number. The first unit must be 0 with no gaps in subsequent numbering. If only one LUN is configured, the LUN number can be omitted

Example:

```
load virtual_scsi_cdrom/chscsi pka_0 scsi_bus=pka scsi_id=0
set pka_0 container="/dev/sr0"
```

This example associates an unallocated CD-ROM drive "/dev/sr0" with virtual_scsi_cdrom "pka_0".

physical_scsi_device

Use "physical_scsi_device" to connect any host "raw" SCSI device to CHARON-VAX. This mapping type is suitable only for tape drives and other specific SCSI devices connected to CHARON host.

"physical_scsi_device" has the following parameters:

container

Parameter	container[N] N=0…7
Type	Text String
Value	A string containing the device name to map to the emulator. ■ **Physical tape drives and other SCSI devices connected to CHARON host** 　■ "/dev/sg<X>", where X is 0, 1, …

media_type

Parameter	media_type[N] N=0…7
Type	Text String
Value	Overrides the PRODUCT ID in the default SCSI INQUIRY data. Valid values may contain uppercase letters, integers and spaces. The length of the string cannot exceed 16 characters. If not specified, synthetic SCSI INQUIRY data is returned containing a PRODUCT ID selected based on the disk size. Initially left unspecified.

geometry

Parameter	geometry[N] N=0…7
Type	Text String
Value	This formatted string value specifies the explicit geometry of the disk storage element The string format is <X>"/"<Y>["/"<Z>] where: ● "X" is the number of sectors per track; ● "Y" is the number of tracks per cylinder; ● "Z" (optional) is the number of cylinders on the unit. If omitted, Z is calculated based on X, Y and the total number of sectors on the unit that reflects the size of the disk storage element; If this parameter is not set, CHARON-VAX will configure the geometry based on the most probable disk type.

use_io_file_buffering

Parameter	use_io_file_buffering[N] N=0…7
Type	boolean
Value	Enables use of the host OS I/O buffering. Initially set to "false" (buffering disabled).

removable

Parameter	removable[N] N=0…7
Type	boolean
Value	Enables the logical unit to appear as a removable SCSI disk drive. Initially set to "false" (fixed, non-removable).

disconnect_timeout

Parameter	disconnect_timeout[N] N=0…7
Type	Numeric
Value	Sets logical unit disconnect timeout. This parameter helps if a connected SCSI device performs a given SCSI command for a very long time. The default value depends on the type of the SCSI device attached. For example for a disk it is 10 seconds, for a tape - an hour. If the type of the device is not known the timeout is 1 hour. **Example with 48 days timeout**: `set pka_0 disconnect_timeout[1] = 0x400000`

⚠ In the table above N stands for logical unit number. The first unit must be 0 with no gaps in subsequent numbering. If only one LUN is configured, the LUN number can be omitted.

Example:

```
load physical_scsi_device/chscsi pka_0 scsi_bus=pka scsi_id=0
set pka_0 container="/dev/sg3"
```

This example associates a host tape device "/dev/sg3" with physical_scsi_device "pka_0".

DSSI Subsystem

Table of Contents

Introduction

The DSSI storage subsystem for the VAX 4000 Models 106, 108, 700 and 705 emulators is based on the emulation of SHAC host adapters and the ability to route SCS cluster information between the emulated SHAC host adapters of multiple nodes via separate TCP/IP links.

The DSSI storage subsystem is functionally emulated, but the emulation is incompatible with the physical DSSI and operates at a much higher throughput than the original hardware. Connection to physical DSSI hardware is neither possible nor planned for future releases.

This version of DSSI emulation for CHARON-VAX supports up to 3 VAX nodes in a virtual DSSI cluster and handles a maximum cluster size of 8 nodes. A single virtual DSSI network supports up to 256 storage elements.

To use a **single** CHARON-VAX system with DSSI emulation, either or both of two elements must be configured:

1. A DSSI storage element (disk or tape).
2. A DSSI storage controller. Currently an emulated HSD50 storage controller is provided. The emulated HSD50 supports physical host drives, CD-ROM drives, physical tapes, removable disks, virtual disks and virtual tapes.

To create a **cluster** of DSSI interconnected CHARON-VAX systems, the DSSI hardware topology is emulated by establishing TCP/IP channels between the emulated SHAC host adapters of each CHARON-VAX system (The use of TCP/IP for the interconnects makes the cluster in principle routable in a WAN). The emulated HSD50 storage controllers are then connected to every SHAC host adapter in the virtual DSSI network.

Cluster operation requires (virtual) disks that are simultaneously accessible by all CHARON-VAX nodes involved. This can be implemented for instance by using a properly configured iSCSI initiator / target structure or a fiber channel storage back-end. Disks on a multiport SCSI switch are not acceptable, as a SCSI switch does not provide true simultaneous access to multiple nodes.

ⓘ When a tape or disk image connected to an emulated DSSI controller is dismounted in VAX/VMS, it is disconnected from CHARON-VAX and can be manipulated. It may even be replaced with a different disk image if it keeps the same name. This capability may be useful when designing back-up and restore procedures.

> The emulated DSSI subsystem has many configurable parameters when multiple nodes on a single DSSI bus are to be connected. Incorrect configuration, in particular non-identical specification of emulated HSD50 disks in the DSSI nodes, is likely to cause data corruption. It is advisable to start any field test by implementing a single node.

Back to Table of Contents

SHAC host adapter

To connect an emulated VAX 4000 model 106, 108, 700 and 705 node to a virtual DSSI network, CHARON-VAX configuration must load at least one emulated SHAC host adapter.

Emulated VAX 4000 models 106, 108, 700 and 705 have two pre-loaded SHAC host adapters named "PAA" and "PAB" There is no need to load any extra instances of SHAC in configuration file.

Note that VAX/VMS running on an emulated VAX 4000 model 106 or 108 node enumerates the emulated SHAC host adapters and assigns them the VMS internal names "PAA" and "PAB". It is recommended for clarity to keep the same naming scheme in the CHARON-VAX configuration file for these emulated SHAC host adapters.

The SHAC emulation has the following configuration parameters:

port

Parameter	port[N] N=0…7
Type	Numeric
Value	An integer value that specifies the TCP/IP port number at which the emulated SHAC host adapter listens for connections from another emulated SHAC host adapter with DSSI node id N. Possible values are from 1024 through 32767. Initially not set.

host

Parameter	host[N] N=0…7
Type	Text String
Value	A string value that specifies the TCP/IP host name (and optionally the TCP/IP port number) to connect to another emulated SHAC host adapter with DSSI node id N. The syntax for the string is "host-name[:port-no]", with possible values for port-no in the range from 1024 through 32767. Initially not set.

scs_node_name

Parameter	scs_node_name[N] N=0…7
Type	Text String
Value	A string value that specifies the SCSNODENAME of the emulated storage element. The string is up to 6 characters long. Possible characters are uppercase letters A through Z , figures 0 through 9. Initially set to an arbitrary value that is guaranteed to be unique within the running emulated VAX 4000 model 106, 108, 700 or 705 node.

scs_system_id

Parameter	scs_system_id[N] N=0…7
Type	Numeric
Value	An integer value that specifies the SCSSYSTEMID of the emulated storage element. Initially set to an arbitrary value that is guaranteed to be unique within the running emulated VAX 4000 model 106, 108, 700 or 705 node.

mscp_allocation_class

Parameter	mscp_allocation_class[N]
	N=0…7
Type	Numeric
Value	An integer value that specifies the ALLOCLASS of the emulated storage element.
	Possible values are from 0 through 255.
	Initially set to 0 which means no allocation class assigned.

container

Parameter	container[N]
	N=0…7
Type	Text String
Value	A string value that specifies the container of the storage element with DSSI node id N and MSCP unit number N. This storage element might be either a (virtual) disk or tape. In VMS running on an emulated VAX 4000 model 106 or 108 node, these storage elements appear as DSSI disks (DIAN:) or DSSI (TF86) tapes (MIAN:).
	Possible values of the parameter are strings in one of the following forms:
	Physical disk drives
	• "/dev/sd<L>", where L is letter
	• "/dev/disk/by-id/..." - addressing by the disk ID, for example "/dev/disk/by-id/ata-ST1000DM003-9YN162_S1D01QJ4"
	• "/dev/disk/by-label/..." - addressing by the disk label, for example "/dev/disk/by-label/MyStorage"
	• "/dev/disk/by-uuid/..." - addressing by the disk UUID, for example "/dev/disk/by-uuid/0e808a2f-cdd3-4944-a245-f729ffd73882"
	Since "/dev/sd<L>" addressing is not persistent, so it is strongly recommended to use "/dev/disk/by-id/wwn-*" syntax instead to refer the disk by its WWID - especially in the enviroments utilizing FC and SAN storages.
	It is also possible to use not a whole disk, but previously created partitions on it. In this case the syntax is the following: "/dev/sd<L><N>" where N is the number of partition to be used.
	Physical tape drives
	• "/dev/sg<X>", where X is 0, 1, …
	Floppy drives
	• "/dev/fd<X>", where X is 0, 1, …
	CD-ROM drives (read-only)
	• "/dev/cdrom<X>" or
	• ("/dev/sr<X>"), where X is 0, 1, …
	Multipath disks
	• "/dev/dm-<X>",
	• "/dev/mapper/mpath<X>",
	• "/dev/mapper/disk<X>", where X is 0, 1, …
	CHARON-VAX disk images
	• [<path-name>"/"]<file-name>[".vdisk"]
	CHARON-VAX tape images
	• [<path-name>"/"]<file-name>".vtape"
	This parameter is initially not set, thus creating NO storage elements on the bus with corresponding DSSI node id.

media_type

Parameter	media_type[N]
	N=0…7
Type	Text String
Value	Overrides default (automatically determined) MSCP media type of the device. Syntax: ```"<device-name>,<device-type>"``` where: • <device-name> is one of "DU", "MU", "DK", "MK", "SCSI", "DI", "MI", "DSSI", "DJ", "MJ" • <device-type> is of the form "LLD" or "LLLD", where "L" is letter from A through Z, and "D" is decimal number from 0 through 99 If not specified, the device name is set to "DI", and the device type is selected based on disk size for disk storage elements. For tape storage elements, the device name and type are set to "MI" and "TF86", respectively. Initialy not specified.

geometry

Parameter	geometry[N]
	N=0…7
Type	Text String
Value	This formatted string value specifies the explicit geometry of the disk storage element with DSSI node id N and MSCP unit number N. This parameter is not applicable to tape storage elements.storage elements. The string format is <X>"/"<Y>["/"<Z>] where: 1. X is number of sectors on track 2. Y is number of tracks on cylinder 3. Z (optional) is the number of cylinders on the unit. If omitted, Z is calculated based on X, Y and the total number of sectors on the unit that reflects the size of the disk storage element If this parameter is not set, CHARON-VAX will configure the geometry based on the most probable disk type. Initially not set. The syntax above is applicable only to disk storage elements. If the container is a tape image, the following format is used instead: Syntax: ```"<image-size>[, <early-warning-zone-size>]"``` where: • <image-size> is the tape size in MB • <early-warning-zone-size> is a size (in KB) of a space left on the tape when a warning to the OS is issued. If omitted, 64K is assumed.

use_io_file_buffering

Parameter	use_io_file_buffering[N] N=0…7
Type	Boolean
Value	Enables use of host OS I/O buffering. Initially set to "false" (buffering disabled).

These parameters are configured with the "set" command.

Example of a Standalone VAX system with 2 disks on a PAA SHAC controller:

```
set session hw_model="VAX_4000_Model_108"
set session log="example1.log"
set toy container="example1.dat"
set rom container="example1.rom"

load operator_console OPA0

set PAA container[0]="/charon/disks/dia0-rz24-vms-v6.2.vdisk"
set PAA container[1]="/charon/disks/dia1-rz24-vms-v6.2.vdisk"
```

The emulated VAX 4000 model 106 or 108 can then boot VMS with the following command:

```
>>> BOOT DIA0
```

After logging into VMS, the "SHOW DEVICE" command displays the following:

```
$ show devices d

Device              Device        Error   Volume        Free  Trans Mnt
 Name               Status        Count    Label       Blocks Count Cnt
004200$DIA0:        Mounted           0   DSSI01        32022   147   1
004201$DIA1:        Online            0
```

HSD50 storage controller

To connect a storage controller to the virtual DSSI network, the CHARON-VAX configuration file must load at least one emulated HSD50 storage controller. In most cases one emulated HSD50 storage controller per virtual DSSI network is enough. The CHARON-VAX configuration file must supply a unique reference name for that instance. Even though this name is only valid within the configuration file, it is recommended for clarity to use the VMS SC SNODENAME as the instance name.

The line below loads an emulated HSD50 storage controller, assigns it the instance name SCSNOD and connects it to the primary built-in DSSI controller:

```
load HSD50 MYDISKS dssi_host=PAA
```

The HSD50 emulation has the following configuration parameters:

dssi_host

Parameter	dssi_host
Type	Text String
Value	A string value that specifies the instance name of an emulated SHAC host adapter serving the virtual DSSI network.
	If this value is not set, CHARON-VAX will try to locate the host adapter automatically. This automatic lookup works only if the CHARON-VAX configuration has exactly one instance of an emulated SHAC host adapter.

dssi_node_id

Parameter	dssi_node_id
Type	Numeric
Value	An integer value that specifies the address of an emulated HSD50 storage controller on the virtual DSSI network.
	Possible values are from 0 through 7 (initially set to 0).

scs_node_name

Parameter	scs_node_name
Type	Text String
Value	A string value that specifies the SCSNODENAME of the emulated HSD50 storage controller.
	The string is up to 6 characters long. Possible characters are uppercase letters A through Z, figures 0 through 9.
	Initially set to the name of the emulated HSD50 controller. Therefore, the name of emulated HSD50 controller should follow the above rules.

scs_system_id

Parameter	scs_system_id
Type	Numeric
Value	An integer value that specifies the SCSSYSTEMID of the emulated HSD50 storage controller.
	Initially set to an arbitrary value that is guaranteed to be unique within the running emulated VAX 4000 model 106, 108, 700 and 705 node.

mscp_allocation_class

Parameter	mscp_allocation_class
Type	Numeric
Value	An integer value that specifies the ALLOCLASS of the emulated HSD50 storage controller.
	Possible values are from 0 through 255 (initially set to 0).

container

Parameter	container[N]
	N=0…9999
Type	Text String
Value	A string value that specifies the container of the storage element with MSCP unit number N. This storage element might be either a (virtual) disk or tape. In VMS running on an emulated VAX 4000 node, these storage elements appear as HSX00 disks (DUAN:) or HST00 tapes (MUAN:).

Possible values of the parameter are strings in one of the following forms:

- **Physical disk drives**
  - "/dev/sd<L>", where L is letter
  - "/dev/disk/by-id/..." - addressing by the disk ID, for example "/dev/disk/by-id/ata-ST1000DM003-9YN162_S1D01QJ4"
  - "/dev/disk/by-label/..." - addressing by the disk label, for example "/dev/disk/by-label/MyStorage"
  - "/dev/disk/by-uuid/..." - addressing by the disk UUID, for example "/dev/disk/by-uuid/0e808a2f-cdd3-4944-a245-f729ffd73882"

  > Since "/dev/sd<L>" addressing is not persistent, so it is strongly recommended to use "/dev/disk/by-id/wwn-*" syntax instead to refer the disk by its WWID - especially in the enviroments utilizing FC and SAN storages.

  ℹ️ It is also possible to use not a whole disk, but previously created partitions on it. In this case the syntax is the following: "/dev/sd<L><N>" where N is the number of partition to be used.

- **Physical tape drives**
  - "/dev/sg<X>", where X is 0, 1, …

- **Floppy drives**
  - "/dev/fd<X>", where X is 0, 1, …

- **CD-ROM drives (read-only)**
  - "/dev/cdrom<X>" or
  - ("/dev/sr<X>"), where X is 0, 1, …

- **Multipath disks**
  - "/dev/dm-<X>",
  - "/dev/mapper/mpath<X>",
  - "/dev/mapper/disk<X>", where X is 0, 1, …

- **CHARON-VAX disk images**
  - [<path-name>"/"]<file-name>[".vdisk"]

- **CHARON-VAX tape images**
  - [<path-name>"/"]<file-name>".vtape"

This parameter is initially not set, thus creating NO storage elements on the controller

media_type

Parameter	media_type[N] N=0…9999
Type	Text String
Value	Overrides default (automatically determined) MSCP media type of the device. Syntax: `"<device-name>,<device-type>"` where: • <device-name> is one of "DU", "MU", "DK", "MK", "SCSI", "DI", "MI", "DSSI", "DJ", "MJ" • <device-type> is of the form "LLD" or "LLLD", where "L" is letter from A through Z, and "D" is decimal number from 0 through 99 If not specified, the device name is set to "DI" and the device type is selected based on disk size for disk storage elements. For tape storage elements, the device name and type are set to "MI" and "TF86" respectively . Initialy not specified.

geometry

Parameter	geometry [N] N=0…9999
Type	Text String
Value	This formatted string value specifies the explicit geometry of the disk storage element with DSSI node id N and MSCP unit number N. This parameter is not applicable to tape storage elements. The string format is <X>"/"<Y>["/"<Z>] where: 1. X is number of sectors on track 2. Y is number of tracks on cylinder 3. Z (optional) is the number of cylinders on the unit. If omitted, Z is calculated based on X, Y and the total number of sectors on the unit that reflects the size of the disk storage element If this parameter is not set, CHARON-VAX will configure the geometry based on the most probable disk type. Initially not set. The syntax above is applicable only to disk storage elements. If the container is a tape image, the following format is used instead: Syntax: `"<image-size>[, <early-warning-zone-size>]"` where: • <image-size> is the tape size in MB • <early-warning-zone-size> is a size (in KB) of a space left on the tape when a warning to the OS is issued. If omitted, 64K is assumed.

use_io_file_buffering

Parameter	use_io_file_buffering[N]
	N=0…9999
Type	Boolean
Value	Enables use of host OS I/O buffering.
	Initially set to "false" (buffering disabled).

Example:

```
load HSD50 DISKS dssi_host=PAA dssi_node_id=5
```

The configuration file below emulates a VAX 4000 Model 108 node, one HSD50 storage controller serving two disks and another instance of an HSD50 controller that serves a tape drive to the VAX over a virtual DSSI:

```
set session hw_model="VAX_4000_Model_108"
set session log="example2.log"
set toy container="example2.dat"
set rom container="example2.rom"

load operator_console OPA0

load HSD50 DISKS dssi_host=PAA dssi_node_id=1

set DISKS container[0]="/charon/disks/dua0-rz24-vms-v6.2.vdisk"
set DISKS container[1]="/charon/disks/dua1-rz24-vms-v6.2.vdisk"

load HSD50 TAPES dssi_host=PAA dssi_node_id=2

set TAPES container[3]="/dev/sg5"
```

In this example we emulate two HSD50 instances. Since they are both connected to the same virtual DSSI bus, we must assign them different DSSI node id values.

The emulated VAX 4000 Model 108 can then boot VMS with the following command:

```
>>> BOOT DUA0
```

After logging into VMS, the "SHOW DEVICE" command displays the following:

```
$ show devices d

Device          Device        Error   Volume        Free   Trans Mnt
 Name           Status        Count   Label        Blocks Count Cnt
DISKS$DUA0:      Mounted         0     DSSI01        31932   147   1
DISKS$DUA1:      Online          0
$ show devices m

Device          Device        Error   Volume        Free   Trans Mnt
 Name           Status        Count   Label        Blocks Count Cnt
TAPES$MUA3:      Online          0
```

CI Subsystem

Table of Contents

Introduction

The virtual CIXCD is the functional equivalent of a hardware CIXCD host adapter, with the exception that there is no physical layer to connect to a hardware CI infrastructure. Since the current host hardware is an order of magnitude faster than the physical CI implementation, such connection - if it were possible - would greatly limit the virtual system throughput.

For data storage, the CIXCD connects to one or more virtual HSJ50 controllers that are loaded as a separate component in the configuration file. To configure VAX CI clusters, the virtual CIXCDs of the multiple CHARON-VAX/66X0 instances are interconnected via TCP/IP links.

Configuring (large) virtual VAX CI clusters requires many configurable parameters and a replicated identical definition of the shared virtual HSJ50 storage controllers in each virtual VAX instance.

To connect a virtual VAX 66x0 to a virtual CI network, the CHARON-VAX configuration file must load at least one virtual CIXCD host adapter; one unit is sufficient in all practical cases.

VAX/VMS enumerates the virtual CIXCD host adapters in the order of increasing XMI node IDs, and assigns them the VMS internal names PAA, PAB, etc. It is recommended for clarity to keep the same naming scheme for virtual CIXCD host adapters in the configuration file.

> The emulated CI subsystem has many configurable parameters when multiple nodes on a single CI bus are to be configured. Incorrect configuration, in particular non-identical specification of the emulated HSJ50 disks in the CI nodes, is likely to case data corruption. It is advisable to start any field test by implementing a single node.

CIXCD host adapter

To connect an emulated VAX 66x0 node to a virtual CI network, the CHARON-VAX configuration must load at least one emulated CIXCD host adapter.

To load the adapter and assign it an instance name of "PAA", enter the following line in the configuration file:

```
load CIXCD PAA
```

The CIXCD emulation has the following configuration parameters:

port

Parameter	port[N] N=0…127
Type	Numeric
Value	An integer value that specifies the TCP/IP port number at which the emulated CIXCD host adapter listens for connections from another emulated CIXCD host adapter with address N. Possible values are from 1024 through 32767. Initially not set.

host

Parameter	host[N] N=0…127
Type	Text String
Value	A string value that specifies the TCP/IP host name (and optionally the TCP/IP port number) to connect to another emulated CIXCD host adapter with address N. The syntax for the string is "host-name[:port-no]", with possible values for port-no in the range from 1024 through 32767. Initially not set.

xmi_node_id

Parameter	xmi_node_id
Type	Numeric
Value	An integer value that specifies the location of the virtual CIXCD host adapter on the XMI bus. Possible values are from 11 through 14 (Initially set to 14).

ci_node_id

Parameter	ci_node_id
Type	Numeric
Value	An integer value that specifies the address of the virtual CIXCD host adapter on the virtual CI network. Possible values are from 0 through 127 (Initially set to 127).

These parameters can be added to the "load" command or specified separately with the "set" command.

The example below shows how to configure a virtual CIXCD adapter with a location on the XMI bus other than the default. It declares a CIXCD adapter in slot 11 (0xB = decimal 11) of the virtual XMI bus:

```
load CIXCD PAA xmi_node_id=0xB
```

The configuration file below creates a virtual VAX 6610 (single CPU) node with one virtual HSJ50 storage controller serving two disks to the VAX over a virtual CI network:

```
set session hw_model="VAX_6000_Model_610"
set session log="vax6610.log"
set toy container="vax6610.dat"
set eeprom container="vax6610.rom"

load operator_console OPA0

load CIXCD PAA

load HSJ50 DISKS

set DISKS container[0]="/charon/disks/dua0-rz24-vms-v6.2.vdisk"
set DISKS container[1]="/charon/disks/dua1-rz24-vms-v6.2.vdisk"
```

HSJ50 storage controller

The virtual HSJ50 storage controller functionally replaces a physical HSJ50 CI storage unit. It supports virtual and physical disks, tapes and removable storage devices that are mapped on local or remote host platform storage. The virtual HSJ50 cannot connect to a physical CI infrastructure.

In a single CHARON-VAX/66X0 instance without a CI cluster, the virtual HSJ50 is located as a separate entity on the same host platform. In a CI cluster, the definition of each HSJ50 is replicated exactly in each CHARON-VAX ~~CI~~ CI node. In most cases one HSJ50 storage controller per virtual CI network is enough.

When loading an instance of a virtual HSJ50 storage controller, the CHARON-VAX configuration file must supply a unique reference name for that instance. While this name is only valid within the configuration file, it is recommended for clarity to use the VAX/VMS *SCSNODENAME* as an instance name.

The line below loads an emulated HSJ50 storage controller and assigns it the instance name SCSNOD:

```
load HSJ50 MYDISKS
```

The HSJ50 emulation has the following configuration parameters:

ci_host

Parameter	ci_host
Type	Text String
Value	A string value that specifies the instance name of the emulated CIXCD host adapter serving the virtual CI network. If this value is not set, CHARON-VAX will try to locate the host adapter automatically. Automatic lookup works only if the CHARON-VAX configuration has exactly one instance of an emulated CIXCD host adapter.

ci_node_id

Parameter	ci_node_id
Type	Numeric
Value	An integer value that specifies the address of an emulated HSJ50 storage controller on the virtual CI network. Possible values are from 0 through 127 (initially set to 0).

scs_node_name

Parameter	scs_node_name
Type	Text String
Value	A string value that specifies the SCSNODENAME of the emulated HSJ50 storage controller. The string is up to 6 characters long. Possible characters are uppercase letters A through Z, figures 0 through 9. Initially set to the name of the emulated HSJ50 controller. Therefore name of emulated HSJ50 controller must follow the above rules.

scs_system_id

Parameter	scs_system_id
Type	Numeric
Value	An integer value that specifies the SCSSYSTEMID of the emulated HSJ50 storage controller. Initially set to an arbitrary value that is guaranteed to be unique within the running emulated VAX 66x0 node.

mscp_allocation_class

Parameter	mscp_allocation_class
Type	Numeric
Value	An integer value that specifies the ALLOCLASS of the emulated HSJ50 storage controller. Possible values are from 0 through 255 (initially set to 0).

container

Parameter	container[N]
	N=0…9999
Type	Text String
Value	Possible values of the parameter are strings in one of the following forms:

Value (continued):

Possible values of the parameter are strings in one of the following forms:

- **Physical disk drives**
  - "/dev/sd<L>", where L is letter
  - "/dev/disk/by-id/..." - addressing by the disk ID, for example "/dev/disk/by-id/ata-ST1000DM003-9YN162_S1D01QJ4"
  - "/dev/disk/by-label/..." - addressing by the disk label, for example "/dev/disk/by-label/MyStorage"
  - "/dev/disk/by-uuid/..." - addressing by the disk UUID, for example "/dev/disk/by-uuid/0e808a2f-cdd3-4944-a245-f729ffd73882"

  > Since "/dev/sd<L>" addressing is not persistent, so it is strongly recommended to use "/dev/disk/by-id/wwn-*" syntax instead to refer the disk by its WWID - especially in the enviroments utilizing FC and SAN storages.

  > ℹ It is also possible to use not a whole disk, but previously created partitions on it. In this case the syntax is the following: "/dev/sd<L><N>" where N is the number of partition to be used.

- **Physical tape drives**
  - "/dev/sg<X>", where X is 0, 1, …

- **Floppy drives**
  - "/dev/fd<X>", where X is 0, 1, …

- **CD-ROM drives (read-only)**
  - "/dev/cdrom<X>" or
  - ("/dev/sr<X>"), where X is 0, 1, …

- **Multipath disks**
  - "/dev/dm-<X>",
  - "/dev/mapper/mpath<X>",
  - "/dev/mapper/disk<X>", where X is 0, 1, …

- **CHARON-VAX disk images**
  - [<path-name>"/"]<file-name>[".vdisk"]

- **CHARON-VAX tape images**
  - [<path-name>"/"]<file-name>".vtape"

This parameter is initially not set, thus creating NO storage elements on the controller

media_type

Parameter	media_type[N]
	N=0…9999
Type	Text String
Value	Overrides default (automatically determined) MSCP media type of the device.
	Syntax:
	```"<device-name>,<device-type>"```
	where:
	• <device-name> is one of "DU", "MU", "DK", "MK", "SCSI", "DI", "MI", "CI", "DJ", "MJ" • <device-type> is of the form "LLD" or "LLLD", where "L" is letter from A through Z, and "D" is decimal number from 0 through 99
	If not specified, the device name is set to "DI" and the device type is selected based on disk size for disk storage elements. For tape storage elements, the device name and type are set to "MI" and "TF86" respectively.
	Initialy not specified.

geometry

Parameter	geometry[N]
	N=0…9999
Type	Text String
Value	This formatted string value specifies the explicit geometry of the disk storage element with CI node id N and MSCP unit number N. This parameter is not applicable to tape storage elements.
	The string format is <X>"/"<Y>["/"<Z>] where:
	1. X is number of sectors on track 2. Y is number of tracks on cylinder 3. Z (optional) is the number of cylinders on the unit. If omitted, Z is calculated based on X, Y and the total number of sectors on the unit that reflects the size of the disk storage element
	If this parameter is not set, CHARON-VAX will configure the geometry based on the most probable disk type.
	Initially not set.
	The syntax above is applicable only to disk storage elements.
	If the container is a tape image, the following format is used instead:
	Syntax:
	```"<image-size>[, <early-warning-zone-size>]"```
	where: • <image-size> is the tape size in MB • <early-warning-zone-size> is a size (in KB) of a space left on the tape when a warning to the OS is issued. If omitted, 64K is assumed.

use_io_file_buffering

Parameter	use_io_file_buffering[N] N=0…9999
Type	Boolean
Value	Enables use of host OS I/O buffering. Initially set to "false" (buffering disabled).

The following example configures a virtual HSJ50 storage controller with a non-default CI network address of 11:

```
load HSJ50 DISKS ci_node_id=0x0B
```

The configuration file below emulates a VAX 6610 node, one HSJ50 storage controller serving two disks:

```
set session hw_model="VAX_6000_Model_610"
set session log=" vax6610.log"
set toy container="vax6610.dat"
set eeprom container="vax6610.rom"

load operator_console OPA0

load CIXCD PAA xmi_node_id=0x0C

load HSJ50 DISKS ci_node_id=0x0B

set DISKS container[0]="/charon/disks/dua0-rz24-vms-v6.2.vdisk"
set DISKS container[1]="/charon/disks/dua1-rz24-vms-v6.2.vdisk"
```

When this configuration file is executed and "container[0]" points to a valid VMS system disk image, a virtual VAX 6610 can boot VAX/VMS with the following command:

```
>>> BOOT /XMI:C /NODE:B DU0
```

In the above boot command, "/XMI:C" and "/NODE:B" instruct the boot ROM to connect to the disk via the host adapter in XMI slot C (the hex value C stands for decimal 12).

Then via the storage controller with CI node id B (decimal 11), DU0 is reached (defined as container[0]) and the boot command is executed for the associated file on the host system.

After logging into VMS, the "SHOW DEVICE" command displays the following:

```
$ show devices

Device             Device        Error    Volume        Free   Trans Mnt
 Name              Status        Count    Label        Blocks Count Cnt
DISKS$DUA0:        Mounted          0     DSSI01        32022   147   1
DISKS$DUA1:        Online           0

Device             Device        Error
 Name              Status        Count
OPA0:              Online           0
FTA0:              Offline          0

Device             Device        Error
 Name              Status        Count
PAA0:              Online           0
```

ℹ When a tape or disk image connected to an emulated HSJ50 controller is dismounted in VAX/VMS, it is disconnected from CHARON-VAX and can be manipulated. It may even be replaced with a different disk image if it keeps the same name. This capability may be useful when designing back-up and restore procedures.

Finding the target "/dev/sg" device

Table of Contents

General description

This section describes how to find proper "/dev/sg" device for CHARON mapping

Procedures of finding the target "/dev/sg" device

First method

In xterm console issue:

```
# cat /proc/scsi/sg/device_hdr; cat /proc/scsi/sg/devices
```

The output will look something like:

```
host    chan    id      lun     type    opens   qdepth  bus     online
4       0       0       0       5       1       1       0       1
5       0       0       0       0       1       1       0       1
```

The fifth field ("type") is the device type.

Value	Device
0	Disk
1	Tape
5	CD-ROM

The "N" in the "/dev/sgN" is the line number in this table (starting from 0) corresponded to the devices CHARON-AXP will use.

Thus "/dev/sg0" will be CD-ROM mapping in this example.

Second method

On a freshly booted system please issue the following command:

```
# dmesg | grep sg
```

The output will look like that:

```
[ 1.503622] sr 4:0:0:0: Attached scsi generic sg0 type 5

[ 1.780897] sd 5:0:0:0: Attached scsi generic sg1 type 0
```

⚠️ This table lists all the devices, not only the real SCSI ones (SATA/IDE for example). CHARON supports only real SCSI devices.

Networking

Table of Contents

Introduction

To configure CHARON-VAX networking, follow these 3 steps:

1. **Load network adapter (if required)**

    If you are configuring a DEQNA, DESQA, DELQA, DEUNA, DELUA, DEMNA, DEBNI or PMADAA adapter, use the "load" command as shown below. CHARON-VAX 3100/96/98 and 4000/106/108 emulations automatically load SGEC (with the name "eza") and therefore no "load" command is required.

    Example:

    ```
    load DELQA/DEQNA NIC
    ```

2. **Load "packet_port" or "tap_port"**

    Load "packet_port" or "tap_port" to connect the SGEC, DEQNA, DESQA, DELQA, DEUNA, DELUA, DEMNA, DEBNI or PMADAA adapter to the host hardware network card (or to a virtual network interface).

    Example:

    ```
    load packet_port/chnetwrk NDIS interface = "eth0"
    ```

3. **Connect the loaded "packet_port" ("tap_port") to the loaded virtual network adapter**

   Connect the SGEC, DEQNA, DESQA, DELQA, DEUNA, DELUA, DEMNA, DEBNI or PMADAA adapter to the "packet_port" ("tap_port") by setting the interface name.

   Example:

   ```
   set NIC interface = NDIS
   ```

The interface name can be either "`(disabled)`" for a disabled interface or "`<network interface name>`"

Examples:

```
load packet_port/chnetwrk NIC1 interface="(disabled)"

load packet_port/chnetwrk NIC2 interface="ens33"
```

SGEC Ethernet Controller

The built-in SGEC controller emulator ("eza") has the following parameters that are specified with the "set" command:

interface

Parameter	interface
Type	Text String
Value	Name of corresponding instance of "packet_port" or "tap_port" component

station_address

Parameter	station_address
Type	Text String
Value	"station_address" provides the ability to configure the adapter's permanent address. By default the adapter's permanent address is read from the host system's NIC.

Set the "station_address" when you need to configure a satellite (remotely booted) system that will run DECnet or when the migrated software uses the permanent address on the network adapter.

Format:

```
XX-XX-XX-XX-XX-XX
```

or

```
XX:XX:XX:XX:XX:XX
```

Example:

```
set eza station_address="AF:01:AC:78:1B:CC"
```

rx_fifo_size

Parameter	rx_fifo_size
Type	Numeric
Value	"rx_fifo_size" sets the receive FIFO size. The value is specified in Kb and by default is pre-calculated from the connected port's size of receive queue. Typically, you do not need to change the "rx_fifo_size" parameter. It is available for extended tuning and bug hunting purposes.

Example:

```
load packet_port/chnetwrk EZA0 interface = "eth0"

set EZA interface = EZA0

set EZA station_address="0C:FE:35:AA:67:3B"
```

DEQNA / DESQA / DELQA Ethernet Controller

CHARON-VAX Q-bus systems provide support for DEQNA, DESQA, DELQA, DEUNA and DELUA Ethernet controllers.

Use the following command to load an instance of DEQNA, DESQA, DELQA, DEUNA and DELUA Ethernet controllers:

```
load DEQNA/DEQNA <logical name>
load DESQA/DEQNA <logical name>
load DELQA/DEQNA <logical name>
load DEUNA/DEUNA <logical name>
load DELUA/DEULA <logical name>
```

Example:

```
load DESQA/DEQNA XQA
```

DEQNA, DESQA, DELQA, DEUNA and DELUA offer the following configuration parameters that can be specified with "set" command:

address

Parameter	address
Type	Numeric

Value	Specifies the CSR address. The address must be a valid QBUS and UNIBUS address in I/O space. Initial value is 017774440 which is the factory setting for DEQNA, DESQA and DELQA Ethernet controllers.
	
	 Use the address parameter if loading several instances of DEQNA, DESQA, DELQA, DEUNA and DELUA.
	
	 "address" parameter value must be unique for every instance of DEQNA, DESQA, DELQA, DEUNA and DELUA.
	
	 Example:
	
	 ```load DEQNA/DEQNA XQA address=017774440```
	```load DEQNA/DEQNA XQB address=017764460```

interface

Parameter	interface
Type	Text String
Value	Name of corresponding instance of "packet_port" or "tap_port" component

station_address

Parameter	station_address
Type	Text String
Value	"station_address" provides the ability to configure the adapter's permanent address. By default the adapter's permanent address is read from the host system's NIC.
	Set the "station_address" when you need to configure a satellite (remotely booted) system that will run DECnet or when the migrated software uses the permanent address on the network adapter.

Format:

```
XX-XX-XX-XX-XX-XX
```

or

```
XX:XX:XX:XX:XX:XX
```

Example:

```
set QNA station_address="AF:01:AC:78:1B:CC"
```

rx_fifo_size

Parameter	rx_fifo_size
Type	Numeric
Value	"rx_fifo_size" sets the receive FIFO size.
	The value is specified in Kb and by default is pre-calculated from the connected port's size of receive queue.
	Typically, you do not need to change the "rx_fifo_size" parameter. It is available for extended tuning and bug hunting purposes.

Example:

```
load DESQA/DEQNA QNA interface = QNA0
set QNA station_address="0C:FE:35:AA:67:3B"
load packet_port/chnetwrk QNA0 interface = "eth0"
```

DEMNA Ethernet Adapter

CHARON-VAX/66X0 systems provide support for the DEMNA Ethernet controller.

Use the following command to load an instance of the DEMNA Ethernet controller:

```
load DEMNA/DEMNA <logical name>
```

Example:

```
load DEMNA/DEMNA EXA
```

DEMNA Ethernet controller offers the following configuration parameters that can be specified with "set" command:

xmi_node_id

Parameter	xmi_node_id
Type	Number
Value	Specifies the XMI slot in which the virtual DEMNA controller is placed. For CHARON-VAX/66X0 a free slot between 10 (A) and 14 (E) must be chosen.

interface

Parameter	interface
Type	Text String
Value	Name of corresponding instance of "packet_port" or "tap_port" component

station_address

Parameter	station_address
Type	Text String

Value	"station_address" provides the ability to configure the adapter's permanent address. By default the adapter's permanent address is read from the host system's NIC.
	
	Set the "station_address" when you need to configure a satellite (remotely booted) system which will run DECnet or when the migrated software uses the permanent address on the network adapter.
	
	Format:
	
	`XX-XX-XX-XX-XX-XX`
	
	or
	
	`XX:XX:XX:XX:XX:XX`
	
	Example:
	
	`set EXA station_address="AF:01:AC:78:1B:CC"`

rx_fifo_size

Parameter	rx_fifo_size
Type	Numeric
Value	"rx_fifo_size" sets the receive FIFO size. The value is specified in Kb and by default is pre-calculated from the connected port's size of receive queue. Typically, you do not need to change the "rx_fifo_size" parameter. It is available for extended tuning and bug hunting purposes.

Example:

```
load DEMNA/DEMNA EXA xmi_node_id = 11 interface = EXA0

set EXA station_address = "0C:FE:35:AA:67:3B"

load packet_port/chnetwrk EXA0 interface = "eth0"
```

DEBNI Ethernet Adapter

CHARON-VAX/63X0 systems provide support for the DEBNI Ethernet controller.

Use the following command to load an instance of the DEBNI Ethernet controller:

```
load DEBNI/DEMNA <logical name>
```

Example:

```
load DEBNI/DEMNA EXA
```

DEBNI Ethernet controller offers the following configuration parameters that can be specified with "set" command:

vax_bi_node_id

Parameter	vax_bi_node_id
Type	Number
Value	Specifies the VAXBI slot in which the virtual DEBNI controller is placed.

interface

Parameter	interface
Type	Text String
Value	Name of corresponding instance of "packet_port" or "tap_port" component

station_address

Parameter	station_address
Type	Text String
Value	"station_address" provides the ability to configure the adapter's permanent address. By default the adapter's permanent address is read from the host system's NIC.
	Set the "station_address" when you need to configure a satellite (remotely booted) system which will run DECnet or when the migrated software uses the permanent address on the network adapter.
	Format:
	``` XX-XX-XX-XX-XX-XX ```
	or
	``` XX:XX:XX:XX:XX:XX ```
	Example:
	``` set EXA station_address="AF:01:AC:78:1B:CC" ```

rx_fifo_size

Parameter	rx_fifo_size
Type	Numeric
Value	"rx_fifo_size" sets the receive FIFO size.
	The value is specified in Kb and by default is pre-calculated from the connected port's size of receive queue.
	Typically, you do not need to change the "rx_fifo_size" parameter. It is available for extended tuning and bug hunting purposes.

Example:

```
load DEBNI/DEMNA EXA vax_bi_node_id = 11 interface = EXA0

set EXA station_address = "0C:FE:35:AA:67:3B"

load packet_port/chnetwrk EXA0 interface = "eth0"
```

PMAD-AA TurboChannel Ethernet Adapter

The CHARON-VAX VAXstation 4000 Model 90 system provides support for a PMAD-AA TurboChannel Ethernet controller (in addition to the preloaded SGEC "EZA").

Use the following command to load an instance of a PMAD-AA Ethernet controller:

```
load PMADAA/PMADAA <logical name>
```

Example:

```
load PMADAA/PMADAA EXA
```

PMAD-AA TurboChannel Ethernet controller offers the following configuration parameters that can be specified with "set" command:

interface

Parameter	interface
Type	Text String
Value	Name of corresponding instance of "packet_port" or "tap_port" component

station_address

Parameter	station_address
Type	Text String

Value	"station_address" provides the ability to configure the adapter's permanent address. By default the adapter's permanent address is read from the host system's NIC.
	
	Set the "station_address" when you need to configure a satellite (remotely booted) system which will run DECnet or when the migrated software uses the permanent address on the network adapter.
	
	Format:
	
	`XX-XX-XX-XX-XX-XX`
	
	or
	
	`XX:XX:XX:XX:XX:XX`
	
	Example:
	
	`set ECA station_address="AF:01:AC:78:1B:CC"`

rx_fifo_size

Parameter	rx_fifo_size
Type	Numeric
Value	"rx_fifo_size" sets the receive FIFO size. The value is specified in Kb and by default is pre-calculated from the connected port's size of receive queue. Typically, you do not need to change the "rx_fifo_size" parameter. It is available for extended tuning and bug hunting purposes.

Example:

```
load PMADAA/PMADAA ECA interface = ECA0

set ECA station_address = "0C:FE:35:AA:67:3B"

load packet_port/chnetwrk ECA0 interface = "eth0"
```

Packet Port

The CHARON-specific "packet_port" interface establishes a connection between an Ethernet adapter in the Linux host system and a network adapter in the virtual VAX system.

For every virtual adapter instance loaded, one dedicated host Ethernet physical adapter is required.

To create instances of "packet_port", use the "load" command in the configuration file as follows:

```
load packet_port/chnetwrk <instance-name>
```

Example:

```
load packet_port/chnetwrk pp_1
```

"packet_port" offers several configuration parameters controlling its behavior.

interface

Parameter	interface
Type	Text string

Value	This parameter Identifies an Ethernet adapter of the host system dedicated to CHARON-VAX.
	
	Syntax:
	
	`set <name> interface="<adapter>"`
	
	Example:
	
	`set pp_1 interface="eth0"`

port_enable_mac_addr_change

Parameter	port_enable_mac_addr_change
Type	Boolean
Value	If "true" is specified, CHARON sets the appropriate Ethernet address automatically. If "false" is specified, set the Ethernet address manually. The default value is "true". **Example**: `set pp_1 port_enable_mac_addr_change=false`

port_retry_on_tx

Parameter	port_retry_on_tx
Type	Numeric
Value	The "port_retry_on_tx" parameter controls the number of times the port will attempt to transmit the packet before giving up. By default, the value is 3. Increasing this value may introduce problems in carrier loosing logic because not all NIC drivers support carrier status query. Typically, you do not need to increase the value. **Example**: `set pp_1 port_retry_on_tx=8`

port_pending_rx_number

Parameter	port_pending_rx_number
Type	Numeric
Value	"port_pending_rx_number" parameter sets the number of pending receive buffers. The default value is 63. The maximum value allowed is 195. You may want to increase the "port_pending_rx_number" when you have very busy networking and experience problems like losing connections not related to the carrier loss. Typically, you do not need to change this parameter. **Example**: `set pp_1 port_pending_rx_number=128`

port_pending_tx_number

Parameter	port_pending_tx_number
Type	Numeric
Value	"port_pending_tx_number" parameter sets the number of buffers the port uses to transmit. The default value is 62. You may want to increase the "port_pending_tx_number" value if the log file indicates dropped TX packets due to TX queue overflow. Typically, you do not need to change this parameter. **Example**: `set pp_1 port_pending_tx_number=128`

log

Parameter	log
Type	Text string

Value	If this parameter is set to some valid file name or a directory where the log files for each individual session will be stored CHARON logs Recv and Xmit packets at the emulated port layer.
	
	If an existing directory is specified, CHARON automatically enables creation of individual log files, one for each session using the same scheme as used for the generation of the rotating log files. If the "log" parameter is omitted, CHARON does not create log.
	
	In certain situations enabling this parameter may help to detect loss of packets.
	
	Example:
	
	`set pp_1 log="pp_1.log"`
	
	`set pp_1 log="/charon/logs"`
	
	Only existing directory can be used as a value of the "log" parameter.

> "tap_port" parameters are the same as "packet_port" ones

Example:

```
load DEQNA/DEQNA XQA

load packet_port/chnetwrk XQA0 interface="eth0"

set XQA interface=XQA0
```

Supported Ethernet Q-bus Adapters for DECnet OSI (DECnet Plus)

The only supported Q-bus device is the DELQA adapter model.

During DECnet OSI installation or DECnet OSI interface reconfiguration, DEQNA and DESQA models are not recognized as valid devices. (DECnet OSI SPD).

> The integrated SGEC Ethernet "EZA" on MicroVAX 3100 & VAX 4000-106/108 is supported by DECnet OSI.
>
> The integrated ESA (AMD Lance 7990) device is not currently implemented in Charon-VAX.

Ethernet Q-bus Adapter and Cluster configuration rules

In a VMS Cluster using VMS V 5.5 and above, use only DELQA and DESQA Ethernet adapters that are supported (VMS Cluster SPD).

The default DEQNA device is not supported for SCS Cluster protocol and the emulated VAX will fail with a CLUEXIT Bugcheck.

Sample configuration files

Contents

VAX 4000 Model 108 configuration file

```
#
# Copyright (C) 1999-2016 STROMASYS
# All rights reserved.
#
# The software contained on this media is proprietary to and embodies
# the confidential technology of STROMASYS. Possession, use, duplication,
# or dissemination of the software and media is authorized only pursuant
# to a valid written license from STROMASYS.
#
#==============================================================================
#
# Sample configuration file for VAX 4000 Model 108.
#
# Specify the hw_model prior to any other commands. This parameter informs
# the emulator what type of VAX it should build and enables all other
# commands.
#
#------------------------------------------------------------------------------

set session hw_model = VAX_4000_Model_108

#==============================================================================
#
# Choose a name for the instance, if needed, to differentiate it among other
# instances running on the same host.
#
#------------------------------------------------------------------------------

#set session configuration_name = VAX_4000_Model_108

#==============================================================================
#
# Use the following commands to disable the rotating LOG files and enable
# a single LOG file. Select either append or overwrite (for each time the
# instance starts) and specify desired log path and file name.
#
#------------------------------------------------------------------------------

set session log_method = append
#set session log_method = overwrite
#set session log = VAX_4000_Model_108.log

#==============================================================================
#
# The following line tells the emulator where to preserve NVRAM content.
# The file maintains the current time of the emulated VAX (when it is not
# running) and other console parameters (such as default boot device).
#
#------------------------------------------------------------------------------

#set toy container="vx4k108.dat"

#==============================================================================
#
# The following line tells the emulator where to store intermediate state
# of the Flash ROM and console parameters. It is highly recommended to enable
# this and the previous toy file for the emulator to be able to correctly
# preserve the saved state of the console.
#
#------------------------------------------------------------------------------

#set rom container="vx4k108.rom"

#==============================================================================
#
# Specify the size of RAM (default is 16MB). Note that the license might
```

```
# limit the maximum amount of memory.
#
#-------------------------------------------------------------------------------

#set ram size=32
#set ram size=64
#set ram size=80
#set ram size=128
#set ram size=256
#set ram size=512


#===============================================================================
#
# Assign four built-in serial lines as necessary. Currently the emulator
# offers two ways to use built-in serial lines:
#
# 1) connection to COM ports (via physical_serial_line) and
# 2) attach a third party terminal emulator (virtual_serial_line).
#
# Once the desired connection is chosen and the corresponding line is
# enabled, connect it to the preloaded controller QUART by chosing the QUART
# line number (in square brackets). See OPA0 example, below.
#
#-------------------------------------------------------------------------------

#load physical_serial_line/chserial TTA0 line="/dev/ttyN"
#load virtual_serial_line/chserial TTA0 port=10000
#set quart line[0]=TTA0

#load physical_serial_line/chserial TTA1 line="/dev/ttyN"
#load virtual_serial_line/chserial TTA1 port=10001
#set quart line[1]=TTA1

#load physical_serial_line/chserial TTA2 line="/dev/ttyN"
#load virtual_serial_line/chserial TTA2 port=10002
#set quart line[2]=TTA2

#===============================================================================
#
# Select the connection method for the console serial line OPA0.
#
#-------------------------------------------------------------------------------

#load physical_serial_line OPA0 line="/dev/ttyN"
#load virtual_serial_line OPA0 port=10003
load operator_console OPA0
set quart line[3]=OPA0

#-------------------------------------------------------------------------------
#
# Uncomment to allow 'F6' to terminate the running emulator.
#
#-------------------------------------------------------------------------------

#set OPA0 stop_on = F6

#===============================================================================
#
# The VAX 4000 Model 108 contains built-in PCI SCSI adapter called PKA
# within the configuration file.
#
#-------------------------------------------------------------------------------
#
# Uncomment to connect the emulator's DKA0 to the disk image.
#
#-------------------------------------------------------------------------------

#load virtual_scsi_disk pka_0 scsi_bus=pka scsi_id=0
#set pka_0 container="<file-name>.vdisk"

#===============================================================================
#
```

```
# Uncomment to connect the emulator's DKA100 to a host disk drive.
#
#-------------------------------------------------------------------------------

#load virtual_scsi_disk pka_1 scsi_bus=pka scsi_id=1
#set pka_1 container="/dev/sd<L>"

#===============================================================================
#
# Uncomment to connect the emulator's GKA200 to an unknown SCSI device.
#
#-------------------------------------------------------------------------------

#load physical_scsi_device pka_2 scsi_bus=pka scsi_id=2
#set pka_2 container="/dev/sg<N>"

#===============================================================================
#
# Uncomment to connect the emulator's DKA300 to the host's CD/DVD-ROM drive.
#
# Device name may be different depending on particular version of host
# operating system. Choose one which suits best.
#
#-------------------------------------------------------------------------------

#load virtual_scsi_cdrom pka_3 scsi_bus = pka scsi_id = 3

#set pka_3 container = "/dev/cdrom"
#set pka_3 container = "/dev/cdrom1"
#set pka_3 container = "/dev/cdrom<N>"
#set pka_3 container = "/dev/sr0"
#set pka_3 container = "/dev/sr<N>"

#===============================================================================
#
# Uncomment to connect the emulator's DKA400 to an .ISO file (CD/DVD image).
#
#-------------------------------------------------------------------------------

#load virtual_scsi_cdrom pka_4 scsi_bus=pka scsi_id=4
#set pka_4 container="<file-name>.iso"

#===============================================================================
#
# Uncomment to connect the emulator's MKA500 to the host's SCSI tape drive.
#
#-------------------------------------------------------------------------------

#load physical_scsi_device pka_5 scsi_bus=pka scsi_id=5
#set pka_5 container="/dev/sg<N>"

#===============================================================================
#
# Uncomment to connect the emulator's MKA600 to a .VTAPE file (tape image).
#
#-------------------------------------------------------------------------------

#load virtual_scsi_tape pka_6 scsi_bus=pka scsi_id=6
#set pka_6 container="<file-name>.vtape"

#===============================================================================
#
# If necessary, load an optional SCSI controller SCSI_B (PKB).
#
# ATTENTION! Old versions of VAX/VMS (older then 5.5-2H4) do not support
# the optional SCSI controller and may fail to boot if this option is loaded.
#
#-------------------------------------------------------------------------------

#include kzdda.icfg

#===============================================================================
#
# Uncomment to enable the built-in SGEC Ethernet Adapter (EZA).
```

```
#
#-------------------------------------------------------------------------------

#set EZA interface=EZA0

#===============================================================================
#
# Connect the SGEC Ethernet Adapter (EZA) to the host's NIC.
#
#-------------------------------------------------------------------------------

#load packet_port EZA0 interface="(disabled)"
#load packet_port EZA0 interface="ethN"

#===============================================================================
#
# Load an optional DHW42-AA (or DHW42-BA, or DHW42-CA) serial line controller
# (C-DAL).
#
# Only one instance of DHW42AA/BA/CA can be loaded.
#
#-------------------------------------------------------------------------------

#load DHW42AA/DHV11 TXA
#load DHW42BA/DHV11 TXA
#load DHW42CA/DHV11 TXA

#load physical_serial_line/chserial TXA0 line="/dev/ttyN"
#load virtual_serial_line/chserial TXA0 port=10010
#set TXA line[0]=TXA0

#load physical_serial_line/chserial TXA1 line="/dev/ttyN"
#load virtual_serial_line/chserial TXA1 port=10011
#set TXA line[1]=TXA1

#load physical_serial_line/chserial TXA2 line="/dev/ttyN"
#load virtual_serial_line/chserial TXA2 port=10012
#set TXA line[2]=TXA2

#load physical_serial_line/chserial TXA3 line="/dev/ttyN"
#load virtual_serial_line/chserial TXA3 port=10013
#set TXA line[3]=TXA3

#load physical_serial_line/chserial TXA4 line="/dev/ttyN"
#load virtual_serial_line/chserial TXA4 port=10014
#set TXA line[4]=TXA4

#load physical_serial_line/chserial TXA5 line="/dev/ttyN"
#load virtual_serial_line/chserial TXA5 port=10015
#set TXA line[5]=TXA5

#load physical_serial_line/chserial TXA6 line="/dev/ttyN"
#load virtual_serial_line/chserial TXA6 port=10016
#set TXA line[6]=TXA6

#load physical_serial_line/chserial TXA7 line="/dev/ttyN"
#load virtual_serial_line/chserial TXA7 port=10017
#set TXA line[7]=TXA7

#===============================================================================
#
# Configure an optional RQDX3 storage controller (MSCP/QBUS). It handles disk
# images, disk drives, CD-ROM drives, magneto-optical drives, floppy drives.
#
#-------------------------------------------------------------------------------

#load RQDX3 DUA

#set DUA container[0]="<file-name>.vdisk"
#set DUA container[1]="/dev/sd<L>"
#set DUA container[2]="/dev/sr<N>"
#set DUA container[3]="<file-name>.iso"

#load RQDX3 DUB address=...
#load RQDX3 DUC address=...
```

```
#============================================================================
#
# Configure an optional TQK50 tape storage controller (TMSCP/QBUS). It handles
# tape images, and physical tape drives attached to the host.
#
#----------------------------------------------------------------------------

#load TQK50 MUA

#set MUA container[0]="<file-name>.vtape"
#set MUA container[1]="/dev/sg<N>"

#load TQK50 MUB address=...
#load TQK50 MUC address=...

#============================================================================
#
# Configuring an optional DELQA Ethernet adapters (QBUS).
#
#----------------------------------------------------------------------------

#load DELQA/DEQNA XQA

#load packet_port/chnetwrk XQA0 interface="ethN"
#set XQA interface=XQA0

#load DELQA XQB address=...
#load DELQA XQC address=...

#============================================================================
#
# Configure an optional DHV11 (or DHQ11, CXY08, CXA16, CXB16) serial line
# controller (QBUS). The address and vector must be set as required by the
# guest operating system.
#
#----------------------------------------------------------------------------

#load DHV11/DHV11 TXA
#load DHQ11/DHV11 TXA
#load CXY08/DHV11 TXA
#load CXA16/DHV11 TXA
#load CXB16/DHV11 TXA

#load physical_serial_line/chserial TXA0 line="/dev/ttyN"
#load virtual_serial_line/chserial TXA0 port=10010
#set TXA line[0]=TXA0

#load physical_serial_line/chserial TXA1 line="/dev/ttyN"
#load virtual_serial_line/chserial TXA1 port=10011
#set TXA line[1]=TXA1

#load physical_serial_line/chserial TXA2 line="/dev/ttyN"
#load virtual_serial_line/chserial TXA2 port=10012
#set TXA line[2]=TXA2

#load physical_serial_line/chserial TXA3 line="/dev/ttyN"
#load virtual_serial_line/chserial TXA3 port=10013
#set TXA line[3]=TXA3

#load physical_serial_line/chserial TXA4 line="/dev/ttyN"
#load virtual_serial_line/chserial TXA4 port=10014
#set TXA line[4]=TXA4

#load physical_serial_line/chserial TXA5 line="/dev/ttyN"
#load virtual_serial_line/chserial TXA5 port=10015
#set TXA line[5]=TXA5

#load physical_serial_line/chserial TXA6 line="/dev/ttyN"
#load virtual_serial_line/chserial TXA6 port=10016
#set TXA line[6]=TXA6

#load physical_serial_line/chserial TXA7 line="/dev/ttyN"
#load virtual_serial_line/chserial TXA7 port=10017
#set TXA line[7]=TXA7
```

```
#load DHV11 TXB address=...
#load DHQ11 TXB address=...
#load CXY08 TXB address=...
#load CXA16 TXB address=...
#load CXB16 TXB address=...

# this is the end of the configuration file ################################
```

Back to Top

VAX 6310 configuration file

```
#
# Copyright (C) 1999-2016 STROMASYS
# All rights reserved.
#
# The software contained on this media is proprietary to and embodies
# the confidential technology of STROMASYS. Possession, use, duplication,
# or dissemination of the software and media is authorized only pursuant
# to a valid written license from STROMASYS.
#
#=============================================================================
#
# Sample configuration file for VAX 6000 Model 310.
#
#-----------------------------------------------------------------------------

set session hw_model = VAX_6310

#=============================================================================
#
# Choose a name for the instance, if needed, to differentiate it among other
# instances running on the same host.
#
#-----------------------------------------------------------------------------

#set session configuration_name = VAX_6310

#=============================================================================
#
# Use the following commands to disable the rotating LOG files and enable
# a single LOG file. Select either append or overwrite (for each time the
# instance starts) and specify desired log path and file name.
#
#-----------------------------------------------------------------------------

set session log_method = append
#set session log_method = overwrite
#set session log = VAX_6310.log

#=============================================================================
#
# The following line tells the emulator where to preserve NVRAM content.
# The TOY file maintains the current time of the emulated VAX (when it is not
# running) and other console parameters (such as default boot device).
#
# Both files must be enabled to correctly preserve the saved state of the console.
#
#-----------------------------------------------------------------------------

#set toy container = "charon.dat"
#set eeprom container = "charon.rom"

#=============================================================================
#
# Specify the size of RAM (default 32MB). Note that the dongle license might
# limit the maximum amount of memory.
#
#-----------------------------------------------------------------------------

#set ram size = 64
#set ram size = 128
#set ram size = 256
#set ram size = 512

#=============================================================================
#
# Select the connection method for the console serial line OPA0.
#
#-----------------------------------------------------------------------------
```

```
#load physical_serial_line OPA0 line = "/dev/tty<N>"
#load virtual_serial_line OPA0 port = 10003
load operator_console OPA0

#-------------------------------------------------------------------------------
#
# Uncomment to allow 'F6' to terminate the running emulator.
#
#-------------------------------------------------------------------------------

#set OPA0 stop_on = F6

#===============================================================================
#
# DWMBB XMI VAXBI Adapter
#
# Slots 1 - 15 (1 - F) of the VAXBI are able to handle I/O adapters.
#
#-------------------------------------------------------------------------------

load DWMBB XBA xmi_node_id = 14

#===============================================================================
#
# KDB50 VAXBI Disk Controller
#
#-------------------------------------------------------------------------------

#load KDB50 DUA vax_bi_node_id = 1

#===============================================================================
#
# Uncomment to connect the emulator's DUA0 to the disk image.
#
#-------------------------------------------------------------------------------

#set DUA container[0] = "<file-name>.vdisk"

#===============================================================================
#
# Uncomment to connect the emulator's DUA1 to the physical disk drive.
#
#-------------------------------------------------------------------------------

#set DUA container[1] = "/dev/sg<N>"

#===============================================================================
#
# Uncomment to connect the emulator's DUA9 to host's CD/DVD-ROM drive or
# to an .ISO file (CD/DVD image).
#
#-------------------------------------------------------------------------------

#set DUA container[9] = "/dev/sg<N>"
#set DUA container[9] = "<file-name>.iso"

#===============================================================================
#
# DEBNI VAXBI Ethernet Adapter
#
#-------------------------------------------------------------------------------

#load DEBNI/DEMNA ETA vax_bi_node_id = 2 interface = ETA0
#load packet_port ETA0 interface = "(disabled)"
#set ETA0 interface = "eth<N>"

#===============================================================================
#
# DWBUA VAXBI UNIBUS Adapter
#
#-------------------------------------------------------------------------------

load DWBUA UBA vax_bi_node_id = 14

#===============================================================================
```

```
#
# TUK50 UNIBUS Tape Controller
#
#-------------------------------------------------------------------------------

#load TUK50 MUA

#===============================================================================
#
# Uncomment to connect the emulator's MUA0 to the physical tape drive or
# to a .VTAPE file (tape image).
#
#-------------------------------------------------------------------------------

#set MUA container[0] = "/dev/sg<N>"
#set MUA container[0] = "<file-name>.vtape"

# this is the end of the configuration file ################################
```

Back to Top

VAX 6610 configuration file

```
#
# Copyright (C) 1999-2016 STROMASYS
# All rights reserved.
#
# The software contained on this media is proprietary to and embodies
# the confidential technology of STROMASYS. Possession, use, duplication,
# or dissemination of the software and media is authorized only pursuant
# to a valid written license from STROMASYS.
#
#===============================================================================
#
# Sample configuration file for VAX 6000 Model 610.
#
#-------------------------------------------------------------------------------

set session hw_model = VAX_6610

#===============================================================================
#
# Choose a name for the instance, if needed, to differentiate it among other
# instances running on the same host.
#
#-------------------------------------------------------------------------------

#set session configuration_name = VAX_6610

#===============================================================================
#
# Use the following commands to disable the rotating LOG files and enable
# a single LOG file. Select either append or overwrite (for each time the
# instance starts) and specify desired log path and file name.
#
#-------------------------------------------------------------------------------

set session log_method = append
#set session log_method = overwrite
#set session log = VAX_6610.log

#===============================================================================
#
# To enable automatic boot, define the default boot device in the VAX
# console and uncomment the line below.
#
#-------------------------------------------------------------------------------

#set xmi boot = auto

#===============================================================================
#
# The following line tells the emulator where to preserve NVRAM content.
# The TOY file maintains the current time of the emulated VAX (when it is not
# running) and other console parameters (such as default boot device).
#
# Both files must be enabled to correctly preserve the saved state of the console.
#
#-------------------------------------------------------------------------------

#set toy container = "charon.dat"
#set eeprom container = "charon.rom"

#===============================================================================
#
# Specify the size of RAM (default 32MB). Note that DIT (when enabled)
# also needs certain amount of memory which grows linearly following
# the size of memory specified here. Also remember that the dongle
# license might limit the maximum amount of memory.
#
# The valid settings are: 32,64,128,256,512,768,1024, ... 3072,
```

```
#
#------------------------------------------------------------------------------

#set ram size = 512
#set ram size = 768
#set ram size = 1024
#set ram size = 2048
#set ram size = 3584

#==============================================================================
#
# Select the connection method for the console serial line OPA0.
#
#------------------------------------------------------------------------------

#load physical_serial_line OPA0 line = "/dev/tty<N>"
#load virtual_serial_line OPA0 port = 10003
load operator_console OPA0

#------------------------------------------------------------------------------
#
# Uncomment to allow 'F6' to terminate the running emulator.
#
#------------------------------------------------------------------------------

#set OPA0 stop_on = F6

#==============================================================================
#
# Uncomment to enable emulation of KDM70 storage controller.
#
#------------------------------------------------------------------------------

#load KDM70 PUA xmi_node_id = 11

#==============================================================================
#
# Uncomment to connect the emulator's DUA0 to the disk image.
#
#------------------------------------------------------------------------------

#set PUA container[0] = "<file-name>.vdisk"

#==============================================================================
#
# Uncomment to connect the emulator's DUA100 to a host disk drive.
#
#------------------------------------------------------------------------------

#set PUA container[100] = "/dev/sd<L>"

#==============================================================================
#
# Uncomment to connect the emulator's DUA300 to the host's CD/DVD-ROM drive.
#
# Device name may be different depending on particular version of host
# operating system. Choose one which suits best.
#
#------------------------------------------------------------------------------

#set PUA container[300] = "/dev/cdrom"
#set PUA container[300] = "/dev/cdrom1"
#set PUA container[300] = "/dev/cdrom<N>"
#set PUA container[300] = "/dev/sr0"
#set PUA container[300] = "/dev/sr<N>"

#==============================================================================
#
# Uncomment to connect the emulator's DUA400 to an .ISO file (CD/DVD image).
#
#------------------------------------------------------------------------------

#set PUA container[400] = "<file-name>.iso"

#==============================================================================
```

```
#
# Uncomment to connect the emulator's MUA600 to a .VTAPE file (tape image).
#
#-------------------------------------------------------------------------------

#set PUA container[600] = "<file-name>.vtape"

#===============================================================================
#
# Support of CI:
#
# Load CIXCD adapter into slot 12 (C) of the XMI.
#
#-------------------------------------------------------------------------------

#load CIXCD PAA xmi_node_id = 12 ci_node_id = 0x01

#===============================================================================
#
# Support of CI:
#
# Connect HSJ50 storage controller to the CIXCD adapter PAA.
#
#-------------------------------------------------------------------------------

#load HSJ50 PUA ci_node_id = 0x0B mscp_allocation_class = 1

#===============================================================================
#
# Uncomment to connect the emulator's DUA0 to the disk image.
#
#-------------------------------------------------------------------------------

#set PUA container[0] = "<file-name>.vdisk"

#===============================================================================
#
# Uncomment to connect the emulator's DUA100 to a host disk drive.
#
#-------------------------------------------------------------------------------

#set PUA container[100] = "/dev/sd<L>"

#===============================================================================
#
# DEMNA XMI Ethernet Adapter EXA
#
#-------------------------------------------------------------------------------

#load DEMNA EXA xmi_node_id = 13 interface = EXA0
#load packet_port EXA0 interface = "(disabled)"
#set EXA0 interface = "eth<N>"

#===============================================================================
#
# DEMNA XMI Ethernet Adapter EXB
#
#-------------------------------------------------------------------------------

#load DEMNA EXB xmi_node_id = 14 interface = EXB0
#load packet_port EXB0 interface = "(disabled)"
#set EXB0 interface = "eth<N>"

# this is the end of the configuration file ################################
```

Back to Top

CHARON-VAX for Linux deinstallation

Deinstallation procedure

To uninstall the CHARON-VAX product:

1. Stop all running CHARON-VAX instances, remove all CHARON-VAX services.

2. Login as "root" user.

3. Issue the following command:

```
# yum remove aksusbd `rpm -q -a | grep charon`
```

Back to Table of Contents

Appendixes

Contents

- glibc.i686 installation without Internet connection
- Configuring devices on the Qbus of a VAX or CHARON-VAX

glibc.i686 installation without Internet connection

Contents

- Description
- Step-by-step guide
    - Introduction
    - Installation steps
        - Mounting the DVD
        - Installation using the rpm utility
        - Installation using the yum utility

Description

The `glibc.i686` package is required for the CHARON-AXP, CHARON-VAX, CHARON-SSP and CHARON-HPA products. When connected to the Internet, and – for Red Hat Enterprise Linux – registered, this package can be installed as described in the product documentation. Usually the following command is used:

```
# yum install glibc.i686
```

When an Internet connection is not available, or if a server running Red Hat Enterprise Linux is not registered, this package can be installed using the operating system installation DVD or ISO file.

This document explains how to install the package from the installation DVD for the following Linux distributions:

- Red Hat Enterprise Linux 6.x and 7.x
- CentOS 7.x - Everything distribution DVD only.
    ⚠ The Standard distribution DVD of CentOS 7.x does not contain the `glibc.i686` package so the Everything distribution DVD is mandatory if no internet connection is available.

ℹ There are several different solutions for installing glibc.i686 without Internet connection. This document explains some of them. For more information, please contact your system administrator or refer to the administrator's guide for the Linux distribution installed on your CHARON host system.

Step-by-step guide

Introduction

Check first if the glibc.i686 package is installed using the following command:

```
# yum list installed glibc.i686
```

If the package is not installed, the command will report the following message (examples given for Red Hat Enterprise Linux 7.2)

```
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager

This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.

Error: No matching Packages to list
```

If the package is installed, the command will report the following message:

```
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager

This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.

Installed Packages

glibc.i686                                    2.17-105.el7                                    @<source>
```

Alternatively, the following rpm command can be used:

```
# rpm -qa | grep glibc | grep i686
```

If the command reports nothing, the package is not installed.

Installation steps

Mounting the DVD

Please either insert the installation DVD in the drive or use an ISO file.

If the DVD is mounted automatically, we recommend to unmount it and mount it manually on a mount point with no spaces in its name (the yum-config-manager utility does not handle spaces correctly).

Example / Red Hat Enterprise Linux 7.2:

```
# df -k

Filesystem            1K-blocks     Used Available Use% Mounted on

/dev/mapper/rhel-root  37300436 4376776  32923660  12% /

devtmpfs                1925960        0   1925960   0% /dev

tmpfs                   1941228       96   1941132   1% /dev/shm

tmpfs                   1941228     9132   1932096   1% /run

tmpfs                   1941228        0   1941228   0% /sys/fs/cgroup

/dev/sda1                508588   160220    348368  32% /boot

tmpfs                    388248       12    388236   1% /run/user/1000

tmpfs                    388248        0    388248   0% /run/user/0

/dev/sr0                3947824  3947824         0 100% /run/media/stromasys/RHEL-7.2 Server.x86_64

# umount /dev/cdrom

# mkdir -p /media/cdrom

# mount -r /dev/cdrom /media/cdrom
```

By default /dev/cdrom is linked to /dev/sr0.

If you have an ISO image of the distribution CD, you can mount it using a loopback device:

```
# mount /path/to/ISO-image.iso /media/cdrom -o loop
```

Installation using the rpm utility

If you use rpm to install the package, any packages on which it depends must be installed manually.

To do so:

1. Switch to the directory containing the packages. This directory depends on your Linux distribution.

   Example:

   ```
   # cd /media/cdrom/Packages/
   ```

2. Locate the target "glibc.i686" package and the package on which it depends:

   ```
   # ls -l glibc*i686* nss-softokn-freebl*
   ```

3. Install the "glibc.i686" package and any others on which it depends (package versions may differ):
   Example / Red Hat Enterprise Linux 7.2:

   ```
   # rpm -i glibc-2.17-105.el7.i686.rpm nss-softokn-freebl-3.16.2.3-13.el7_1.i686.rpm
   ```

   Example / CentOS7:

   ```
   # rpm -i glibc-2.17-157.el7.i686.rpm nss-softokn-freebl-3.16.2.3-14.4.el7.i686.rpm
   ```

   > - If the command returns a warning message related to RSA/SHA256 signature, please ignore it. The successful installation of the package can be checked using the "yum list installed glibc.i686" command.
   >
   > - If the installation command above reports additional unsatisfied dependencies, add the corresponding packages to the above command line.

4. Umount the CD-ROM or ISO file if necessary:

   ```
   # cd -
   # umount /media/cdrom
   ```

Installation using the yum utility

The yum utility will check and install all the necessary dependencies.

Define the operating system installation DVD as a new repository:

Example / Red Hat Enterprise Linux 7.2:

```
# yum-config-manager --add-repo=file:///media/cdrom

Loaded plugins: langpacks, product-id
adding repo from: file:///media/cdrom

[media_cdrom]
name=added from: file:///media/cdrom
baseurl=file:///media/cdrom
enabled=1

# yum --nogpgcheck install glibc.i686

...
Is this ok [y/d/N]: y

...
Complete!
```

Once the installation is complete, the repository can be disabled (if no other package has to be installed):

```
# yum-config-manager --disable media_cdrom
```

Using this method, installing other packages could require the gpg check to be disabled. To do so:

- either add the `--nogpgcheck` parameter to the `yum install` commands.
  Example:

  ```
  # yum --nogpgcheck install <package>
  ```

- or disable the GPG check for the repository in the `/etc/yum.repos.d/media_cdrom.repo` file, by adding the `gpgcheck=0` line

- or disable the GPG check in the `/etc/yum.conf` file, replacing the `gpgcheck=1` line by `gpgcheck=0`

Example / CentOS 7:

```
# yum --disablerepo=* --enablerepo=c7-media install glibc.i686

...
Is this ok [y/d/N]: y

...
Is this ok [y/d/N]: y

...
Complete!
```

The `c7-media` repository is a default repository on CentOS 7.x pointing to the `/media/cdrom` folder. Please refer to the administrator's documentation for more information.

Configuring devices on the Qbus of a VAX or CHARON-VAX

Related products	CHARON-VAX all versions
Operating systems	Windows, Linux

Table of contents

Description

ⓘ This document corresponds to the AN-032.pdf file, dated September 3rd, 2007

When implementing CHARON-VAX, it may not be possible to implement exactly the same configuration as was used on the original hardware VAX.

In case of difficulty, configure CHARON-VAX by specifying the exact addresses and vectors where necessary in the CHARON configuration file (for CHARON for Windows version 4.8 and later use "Edit Configuration" button of the "CHARON VM Manager" to get access to the configuration file).

For details on the configuration of peripheral device addresses and vectors refer to the VAX peripheral option manuals.

If you move a Qbus system disk from an existing system to an appropriate CHARON-VAX without modification, you must identify the device / controller addresses and vectors and set them to the same value as your VAX operating system used before. For instance, in VMS you can use the MCR SYSGEN (or MCR SYSMAN) and the show / configuration command to identify the values.

But if you are reconfiguring a VAX fortunately, CHARON-VAX can be used to calculate these addresses as follows:

Step-by-step guide

Decide what kinds of QBUS devices are needed and how many of them should be present.

For example, suppose four (4) RQDX3 controllers, two (2) DELQA Ethernet controllers, and three (3) DHV11 controllers are needed. Note that when using a CHARON-VAX TMSCP controller it is seen by the VAX as a TQK50.

For the next step configure CHARON-VAX with only a console terminal. Start CHARON, wait until the ">>>" prompt is displayed, enter the "configure" command and enter you configuration followed by "exit":

```
KA650-A V5.3, VMB 2.7
Performing normal system tests.
40..39..38..37..36..35..34..33..32..31..30..29..28..27..26..25..
24..23..22..21..20..19..18..17..16..15..14..13..12..11..10..09..
08..07..06..05..04..03..
Tests completed.
>>>_
>>>CONFIGURE
Enter device configuration, HELP, or EXIT
Device,Number? HELP
Devices:
 LPV11 KXJ11 DLV11J DZQ11 DZV11 DFA01
 RLV12 TSV05 RXV21 DRV11W DRV11B DPV11
 DMV11 DELQA DEQNA DESQA RQDX3 KDA50
 RRD50 RQC25 KFQSA-DISK TQK50 TQK70 TU81E
 RV20 KFQSA-TAPE KMV11 IEQ11 DHQ11 DHV11
 CXA16 CXB16 CXY08 VCB01 QVSS LNV11
 LNV21 QPSS DSV11 ADV11C AAV11C AXV11C
 KWV11C ADV11D AAV11D VCB02 QDSS DRV11J
 DRQ3B VSV21 IBQ01 IDV11A IDV11B IDV11C
 IDV11D IAV11A IAV11B MIRA ADQ32 DTC04
 DESNA IGQ11
Numbers:
 1 to 255, default is 1
Device,Number? RQDX3,4
Device,Number? DELQA,2
Device,Number? DHV11,3
Device,Number? exit
Address/Vector Assignments
-774440/120 DELQA
-774460/300 DELQA
-772150/154 RQDX3
-760334/304 RQDX3
-760340/310 RQDX3
-760344/314 RQDX3
-760500/320 DHV11
-760520/330 DHV11
-760540/340 DHV11
>>>_
```

It is possible to see addresses and vectors, which need to be put into the configuration file, for each of the required devices.

1. Some QBUS devices are not able to set an interrupt vector. For such Devices, VMS sets the vector programmatically when initializing the device. For example, RQDX3 and DELQA/DEQNA
2. The addresses above are 16-bit wide, but CHARON is able to understand 22-bit addresses. Therefore, it is necessary to sign-extend them before writing into CFG file.
3. Numbers are octal, so attach "0" in the beginning.

For this example, the configuration file would be as follows:

```
load RQDX3/RQDX3 DUA address=017772150
load RQDX3/RQDX3 DUB address=017760334
load RQDX3/RQDX3 DUC address=017760340
load RQDX3/RQDX3 DUD address=017760344
load DELQA/DEQNA XQA address=017774440
load DELQA/DEQNA XQB address=017774460
load DHV11/DHV11 TXA address=017760500 vector=0320
load DHV11/DHV11 TXB address=017760520 vector=0330
load DHV11/DHV11 TXC address=017760540 vector=0340
```

Units must obviously be configured, but it is out of scope here.

Your copy of your VMS system may not automatically understand this configuration. It may be necessary to build a new VMS system from the original VMS media but if all the required elements are available in your VMS system, it may be possible to reconfigure by executing the following commands:

```
$ SET DEF SYS$UPDATE
$ AUTOGEN GETDATA REBOOT
```

This application note is provided for information only to assist customers in dealing with complex configurations. This functionality described is standard VAX and VAX/VMS functionality and is not covered by CHARON-VAX support contracts.

Related articles

- CHARON on Windows - Automated License Expiration Check
- CHARON Log monitoring on Windows (logmond) - Best practices
- Configuring devices on the Qbus of a VAX or CHARON-VAX
- CHARON licensing options
- hasp_srm_view utility