Programme: VLT

Project/WP: VLT Software Infrastructure Maintenance

VLT common software, installation manual

Document Number: ESO-043185

Document Version: 18

Document Type: Manual (MAN)

Released On: 2021-10-11

Document Classification: ESO Internal [Confidential for Non-ESO Staff]

Prepared by: Sylvie Feyrin

Validated by:

Approved by:

Name

Authors

Name	Affiliation
Sylvie Feyrin	ESO

Change Record from previous Version

Affected Section(s)	Changes / Reason / Remarks
All	Upgraded to VLT2019
All	Upgraded to VLT2020

Contents

1. Introduction

1.1 Purpose

This manual provides the installation procedure of the VLT Common Software.

This document assumes that you have an overview of the VLT Common Software, a good LINUX knowledge and some experience in installing public domain software. Unless you are the system manager, you may need his help to perform some of the installation steps.

1.2 Scope

This document describes the installation of the VLT2020 version of the VLT Common Software. The detailed scope is given is section 3.3.

1.3 Definitions, Acronyms and Abbreviations

This document employs several abbreviations and acronyms to refer concisely to an item, after it has been introduced. The following list is aimed to help the reader in recalling the extended meaning of each short expression:

CCS	Central Control Software
CCSlite	VCS providing a subset of functions (no RTAP needed)
DBMS	Database Management System
Full CCS	VCS providing all functions (needs HP/RTAP)
HOS	High Level Operations Software
HW	Hardware
ICS	Instrument Control Software
I/O	Input/Output
LAN	Local Area Network
LCC	LCU Common Software
LCU	Local Control Unit
N/A	Not Applicable
NFS	Network File System
NOCCS	VCS in a minimal configuration
OS	Operating System
OSB	Observation Software base
PCF	Point Config File
ROS	Remote Operation Software
RTAP	Real-Time Application Platform (from Hewlett-Packard)
SPR	Software Problem Report
SW	Software
TBC	To Be Confirmed
TBD	To Be Defined
TCS	Telescope Control Software
UIF	(Portable) User Interface (Toolkit)

VCS	VLT Common Software
VLTSW	synonym of VCS
VLT	Very Large Telescope
WAN	Wide Area Network
WS	Workstation
WSF	Workstation Software Framework

1.4 GLOSSARY

N/A

1.5 STYLISTIC CONVENTIONS

The following styles are used:

`teletype`

in the text, names of programs, files, directory, etc.

in installation examples, your typing.

`<...>`

for parts that have to be substituted with the real content before typing.

bold and *italic* are used to highlight words.

2. Related Documents

2.1 Applicable Documents

N/A

2.2 Reference Documents

In addition to documents that are part of the VLT Software documentation kit (see section 3.3.2), the following documents are referenced in this document.

The following documents, of the exact version shown herein, are listed as background references only. They are not to be construed as a binding complement to the present document.

RD1 VLT Common Software - CentOS Installation Manual;
VLT-MAN-ESO-17200-2009 Version 18

RD2 VLT Electronic Design Specification;
VLT-SPE-ESO-10000-0015 Version 6

3. Overview

3.1 General

Since VLT2011, CCSLite is the only "flavour" for building VLTROOT.

For VLT2020 CentOS 7.7 (64bits) is the official platform.

The version of Linux gcc compiler is 7.3.1 and VxWorks 6.9 with gcc 4.3.3.

Since VLT2015, VLTROOT is compiled for 64bits.

The VLT Common Software currently includes:

- VLT development utilities
- Workstation Software Framework 2 (wsf2)
- Central Common Software (CCS)
- Instrumentation Common Software (INS)
- Telescope Control Software (simulation)
- Sequencer
- WS-LCU communication software (Qserver)
- LCU Common Software (LCC)
- Drivers
- Motor Control
- LCU contributions
- CCD Control Software
- New General detector Controller (NGC)
- BOSS Base Observation Software
- OSB Supplementary modules for BOSS
- Common Library for Image Processing (CLIP)
- Matlab Interface (mlapi)
- TDCS
- installation and installation verification procedures
- simple examples

and all the relevant documentation in portable document format (PDF).

The present version of the VLT Common Software replaces entirely any previous ESO distribution of VLT Software.

The VLT Common Software is distributed together with the Linux OS on a DVD for an easier installation. An additional DVD is provided for VxWorks installation.

For a complete description of each module mentioned, please refer to the appropriate user manual.

The major differences with the previous version as well as currently known problems and workarounds are summarized in the RELEASE NOTES, published on the web together with the documentation.

Even if you are interested in one part only of the VLT Software, please read ALL of the following sections.

3.1.1 Copyright

The VLT Common Software is distributed outside ESO for the development of applications related to the VLT Project and ruled by the “General Conditions of ESO Contracts”. Any other use is not permitted without prior authorization from ESO.

The rights of Third Parties (Free Software Foundation, VxWorks, etc.), whose software is for convenience copied on the DVD, are ruled by their copyright notice included in their software.

The file COPYRIGHT, available together with the documents, contains the full copyright notice.

3.2 Supported Configuration

3.2.1 Hardware

WS

- Dell Poweredge

See the tested configurations in the document [RD1].

LCU:

CPU boards:

- CPU board (68K) Motorola MVME 167 (obsolete but still supported)
- CPU board (Power PC) Motorola MVME 2604
- CPU board (Power PC) Motorola MVME 2700
- CPU board (Power PC) Motorola MVME 6100

Please refer to the detailed list of LCUs in the section 4.3.5.1 of the document [RD2].

3.2.2 Software

UNIX Operating System:

Linux: CentOS 7.7 64bits

Graphical Libraries: X11R6 and Motif2.1.

VxWorks:

- VxWorks 6.9.4 - Wind River Workbench 3.3 for PPC and NEHALEM LCUs

3.3 Contents

The two DVD ISO images of the VLT Common Software can be downloaded from the web page.

The first one contains CentOS 7.7 distribution and the VLT Common Software, and the second one VxWorks 6.9.

For convenience, some public domain software, used in the generation of the VLT Software, are also distributed.

The VMware image is also available and can be downloaded from the web page.

The VLT Common Software documentation kit is available on the web.

3.3.1 Software

1. PUBLIC DOMAIN SOFTWARE
   a) gcc version 7.3.1 (RedHat DevToolset 7)
   b) GNU Make (3.82)
   c) GNU awk (4.0.2)
   d) GNU gdb (7.6.1)
   e) GNU flex (2.5.37)
   f) GNU bison (3.0.4)
   g) GNU gzip (1.5)
   h) GNU sed (4.2.2)
   i) GNU diffutils (3.3)
   j) GNU rcs (5.9.0)
   k) GNU ddd (3.3.12)
   l) GNU zip (3.0)
   m) GNU unzip (6.0)
   n) GNU tar (1.26)
   o) GNU M4 (1.4.16)
   p) GNU autoconf (2.69)
   q) GNU grep (2.20)
   r) GNU gnuplot (4.6.2)
   s) GNU gsl (1.15)
   t) GNU groff (1.22.2)
   u) GNU perl (5.16.3)
   v) Tcl 8.5.13
   w) Tk 8.5.13
   x) [incr Tcl] 3.4 2008-02-07

y) iwidgets 4.0.2 2007-06-10
z) tclX 8.4 2007-02-27
aa) BLT 2.5 patched
bb) msql 2.0.11
cc) tkimg 1.4
dd) Tktable 2.9
ee) snack 2.2.10
ff) tclCheck 1.1.13
gg) expect 5.45
hh) rman 3.1
ii) tkman 2.2
jj) tkinspect 5.1.6p10
kk) tklib 0.5
ll) Sybtcl 2.5
mm) gplot 5.1
nn) TclPro (procheck) 1.5.0
oo) msqltcl (1.99)
pp) tcllib (1.10)
qq) java-1.8.0-openjdk
rr) eclipse Mars
ss) doxygen 1.8.5
tt) dot 2.30.1
uu) cftisio 3.31
vv) fftw (2.1.5 and 3.3.3)
ww) svn client 1.9.5

2. VLT SOFTWARE

UNIX:

a) VLT Utilities

installation scripts (vltsw)
Kit/VLT Makefile and Man-page browser (vlt)
document development support toolkit (doc)
code and document templates (templates)
standard environment (pecsvltsw)
emacs customization kit (emacs)

tool for automatic test (tat)

Workstation Software Framework2 (wsf2)

b) Queue Server Emulator (qsemu)

c) Central Common Software:

CCS/Ccs
CCS/alrm
CCS/cai
CCS/ccs
CCS/ccsMon
CCS/ccsei
CCS/cmd
CCS/db
CCS/dblpp
CCS/eccs
CCS/envs
CCS/err
CCS/evh
CCS/evhEt
CCS/evt
CCS/fnd
CCS/his
CCS/log
CCS/msg
CCS/plot
CCS/samp
CCS/scan
CCS/tims
CCS/vcc
CCS/vccmake
PANEL/fedit
PANEL/panel
PANEL/ptlib
PANEL/uif
PANEL/wdgpan

d) HOS (High-level Operaton Software)

sequencer (seq)
access configuration and control (acc)
Audio software (audio)

e) INS Common Software - File handling

brooker for observing blocks (bob)
SLX/miscellaneous utilities (misc)
SLX/setup file handling (slx)
SLX/setup file handling - C++ version (oslx)
SLX/setup file handling - tcl interface (slxtcl)
INS/data transfer module. (dxf)
INS/ins system tools (ist)
INS/protocol converter. (pco)
INS/common templates (insc)
INS/VLT OnLine Archive interface (volac)
INS/osf   operator support functions

f) Common Library for Image Processing (CLIP)
g) ICB INS Common Base ICS (auto, ctoo, egen, ic0, ic0dev, ic0dig, ic0lcu, ic0mot, ic0sen, icb, icbpan, lccdev, lcctoo, uvisen)
h) Fieldbus Extension
i) CCD Control Software
j) OSB (boss, ibac, ixac, osb)
k) New General detector Controller (NGC)
l) TCS (Telescope Control Software) interface
m) TCS simulation kit (msw, prs, tcs, tcssim, tif, trkws)
n) Real-Time Display (rtdcore, rtdc) and catalog library (astrocat)
o) DICB FITS keyword Dictionaries (dicPAF, dicFITS, dicDPR, dicOBS, dicTPL, dicGEN)
p) Matlab interface (mlapi)
q) an example of a WS application (examples/wsapp)

VxWorks:

a) LCU Common Software

LCC library (lcc3)
Command Interpreter tools (too)

b) LCU QServer (lqs)
c) Drivers

driver log utility (lculog)
driver common utilities (lcudrv)
Time Board driver (tim)
Serial I/F driver (iser)
Can driver (canio)
MEN analog/digital I/O (mendrv)
LCU configuration tool (lcuboot)
LCU status monitoring tool (lcustat, lcuwd)
driver engineering interface basic tool (inducer)
Time Board engineering interface (timx)
canio engineering interface (caniox)

d) Motion Control

Motion Controller wrapper (mcm)
Motor Control Module API (mot)
Motor Command Interface (motci)
Motor Engineering Interface (motei)
SDL Common Interface (sdl)

e) an example of an LCU application (examples/lcuapp)

3.3.2 Documentation

The documentation is organized in the following parts:

1. VLT Core and TCS Interface
2. INS Core and NGC

Documents are provided for external releases directly on the web: https://www.eso.org/sdd/bin/view/SDDPublic/VltDocumentationKit2020

3.4 Backward Compatibility

The software included in the present delivery is normally backward compatible with the previous versions that have been already declared as baseline. For what is concerning software that appear for the first time with this release, please refer to the appropriate User Manual to see whether it is released as a baseline version (i.e., backward compatible in future versions) or still in a preliminary form (i.e. it may change in future versions)

In any case, after the installation of the current release, any existing software/application using the VLTSW must be regenerated. INTROOTs (if any) should be regenerated from scratch.

The “Release notes” point out the differences, between the current version and the previous one and backward incompatibilities, if any.

3.5 Default Shell

Starting with the MAR2001 release, the default shell is bash. We re-wrote also the standard environment, now called PECS (see section 4.2 and A) in order to use bash. Since then, the general organization of the environment, the installation procedure and setting the environment for a user have changed. The implications of this are:

- the account pecsmgr is needed, that is the owner of some core PECS files;
- the users vltmgr and vlt must have the default shell set to /bin/bash;
- vx will remain to /bin/csh
- any other user must be defined with default shell /bin/bash
- the syntax in the bash shell is different (see A).

3.6 Problem Reporting/Change Request

Please use the tool JIRA (https://jira.eso.org) and the JIRA project VLTSW.

4. INSTALLATION

The purpose of this installation manual is to guide the process of installing the VLT Software. Please read this document carefully and follow the instructions provided. This will help you save time and get a better understanding of the VLTSW. Please report errors, misprinting and suggestions to improve this document (see 5.8).

The DVDs for the VLT2020 release of the VLT software performs the installation of the CentOS 7.7 distribution, the complete VLT Common Software, as well as the required external products.

Two DVD are distributed, one with OS and VLTROOT and the second one with VXWORKS when needed.

The first DVD provides an automatic installation procedure, starting with the installation of the operating system. This procedure covers all necessary steps installing the binaries from the DVD and configuring the system.

Please refer to the Linux Installation Manual [RD1] for the automatic installation. When the software is installed please continue with the following section 5 to verify your installation.

Section 5 provides an overall verification test. The verification test is part of the installation. It provides a tutorial on how to configure both WS and LCU, so its execution is recommended. Section 6 provides a trouble- shooting guide.

The following sections give some details about the installation.

4.1 OPERATING SYSTEM

4.1.1 OS

For the VLT2020 release, the DVD installs the CentOS 7.7 Operating System

A monitoring tool called ccsMonGui provides information about the system where the VLTSW is installed. ccsMonGui allows (among other features) to check the machine's kernel parameters.

4.1.2 Users

The following users, belonging to the same group as any other developer, are created during the automatic installation:

`vltmgr`

This user should be the only one authorized to make a new installation of any VLT software in the VLT root area. This user can be also used to perform the verification process.

`vlt`

used to run the default environment and the msqld daemon at boot time and the logging system clean-up procedure. This user is local to each machine and has /`vlt/vlt` as `$HOME`.

`vx`

: used by LCU to access VxWorks kernel at boot time. It shall contain only a `.rhosts` file that allows access to LCU node(s). This user is local to each machine and has `/vlt/vx` as `$HOME`.

`pecsmgr`

: its home directory (that must be /etc/pecs) is the repository for everything concerning the standard environment (see 4.2).

`rtap`

: it is the RTAP/PLUS administrator. For backward compatibility, it is created on a CCSLite machine too.

4.1.3 Disk Areas

The VLTSW uses different areas, created during the automatic installation:

`/vlt`	to store binaries (products & VLT SW)
`/vltdata`	to store ENVIRONMENTS related data

Those areas are implemented as separate logical volumes (on the same or on different physical disks). The layout used can be found in [RD1].

4.2 ENVIRONMENT SETTING

4.2.1 PECS

The standard environment consists of a set of environment variables (PATH, MANPATH, etc). To be sure that the environment is always properly defined, a standard set of files is provided.

The basic standard environment with all the necessary variables is already available after the automatic installation (see also [RD1]).

This environment has to be customised. For this purpose a configuration file is supplied:

/etc/pecs/releases/000/etc/locality/apps-`‘hostname‘.env`

Now:

a. edit your configuration file according to the instruction given in the file itself.
   In particular add the following line at the very bottom of apps-‘hostname‘.env if necessary

```
export SOFTWARE_ROOTS=
export ACC_HOST=‘hostname‘
```

b. the account of the users vltmgr and vlt are configured during the automatic installation. For any new users you need to :
   a. login as that user
   b. execute the following:

```
$ /etc/pecs/bin/pecssh mklinks -i
    PECS_ROOTDIR [/etc/pecs]: "return"
```

```
PECS_RELEASE [000]: "return"
[...]
Do you wish to install VUE support files? [y]: "n"
```

c. logout and login again (if you are using the CDE, exit and re-login)

PECS allows customization at user level, whenever the user wants to change any of the default settings. For example, if you want to change the VLTROOT definition, edit the file

$HOME/.pecs/apps-all.env

The new VLTROOT will be valid for that user on all the machines he logs in (provided the home directory is exported on more than one machine). If you want to change the VLTROOT on a specified machine texx, put the new VLTROOT setting in the file:

$HOME/.pecs/apps-texx.env

More information can be obtained in Appendix A

4.2.2 VLT Root and VLT Data Areas

With the automatic installation only CCSLite VLT Root is now provided.
CCSlite contains all the functionalities of the CCS (based upon a proprietary implementation of a subset of the RTAP software).

The VLT Root area is where binaries, includes, scripts, etc. of the VLT Common Software are located. This area is normally indicated by the environment variable VLTROOT and it is located under the /vlt directory:

```
/vlt/<RELEASE>/CCSlite
```

where <RELEASE> is a directory whose name should be e.g. FEB2000 if you are installing the FEB2000 Release and so on. This directory <RELEASE> is created under /vlt during the PECS installation.

The VLT data area is where the environments, configuration and temporary data are located. This area is normally indicated by the environment variable VLTDATA and it is located under:

`/vltdata` for all installation types

VLTROOT and VLTDATA are defined in /etc/pecs/releases/000/etc/locality/apps-`‘hostname‘.env`.

These directories are populated as vltmgr during the automatic installation.

4.3 PRODUCTS

4.3.1 GNU tools

The VLT Software project has adopted the GNU tools as standard development tools (make, compiler, debugger, etc. See 3.3.1 for the complete list).

Since VLT2017, the default GNU tools delivered by the OS System are used.
Only the missing libraries and scripts are under the dedicated directory defined by the GNU_ROOT environment variable.

The standard environment provides the following definition:

```
$ export GNU_ROOT=/vlt/<RELEASE>/gnu
```

And GNU_ROOT is added in the variables PATH, MANPATH and LD_LIBRARY_PATH.

4.3.2 Tcl/Tk

The basic Tcl/Tk and the extensions are required by several components of the VLT Software.

All the required libraries and scripts in directories under a dedicated directory defined by the TCLTK_ROOT environment variable.

The standard environment provides the following definition:

```
$ export TCLTK_ROOT=/vlt/<RELEASE>/tcltk
```

And TCLTK_ROOT is added in the variables PATH, MANPATH and LD_LIBRARY_PATH.

4.3.3 VxWorks

With VLT2020, VxWorks version 6.9 is provided with Wind River Workbench 3.3.

If you want to generate the LCU software, a fully installed VxWorks environment is required.

VxWorks is a licensed product of WindRiver. If you need a license please contact vltsccm (vltsccm@eso.org).

VxWorks 6.9 and the Workbench are using a license server, which is running on te13.hq.eso.org. So you will need access to the network.

Remember that the VxWorks directory shall be read-accessible by the LCU, i.e., the file downloaded at boot time is readable by the username (vx) used by the LCU to access the host system.

To use VxWorks, each user needs the environment variable setup as described in the installation procedure and provided by the standard environment.

4.3.4 JAVA

JAVA is installed under a dedicated directory defined by the JAVA_HOME environment variable.

The standard environment provides the following definitions:

```
    $ export JAVA_HOME=/vlt/<RELEASE>/jdk1.8.0_77
```

The variables PATH is updated accordingly.

4.3.5 DFS tools (if applicable)

The DFS tools (vcsolac, Fits-header test-tool) are installed under a dedicated directory /vlt/<Release>/DFS

To use the DFS tools, the $DFS_ROOT variable should be defined as:

```
$DFS_ROOT=/vlt/<Release>/DFS
```

4.3.6 CPL libraries

A binary distribution of CPL libraries is installed under a dedicated directory defined by the CPL_HOME environment variable.

The standard environment provides the following definitions:

```
    $ export CPL_HOME=/vlt/<RELEASE>/cpl
```

The variables PATH, MANPATH and LD_LIBRARY_PATH are updated accordingly.

5. VERIFICATION

If you are already familiar with the VLT Common Software, simply browse through section 5.4 to get familiar with any new tool introduced by this version.

This section describes how to setup a simple application environment. Here the most important features of the VLT Common Software are exercised. Thus two objectives are achieved:

- verifying the correctness of the installation you have just completed
- providing a simple tutorial on the use of the VLT Common Software

Detailed information on how to configure each module is provided by the appropriate User Manual.

The scope of this section is limited to a WS and a LCU in a one-environment-per-node configuration. For the configuration and verification of Drivers and Motor Control and for more environments on a WS, please refer to the User Manuals.

The following sections assume that:

- you are familiar with configuring UNIX and, as appropriate, VxWorks utilities and that node names, IP addresses, etc. are already correctly configured on your WS.
- you have one WS with CCSLite and one LCU and you set up an environment on each. In this chapter `<wsenv>` and `<lcuenv>` will be used to represent the name of the two environments.

Generally speaking, in order to verify the correct file setup, configuration activities should be done by a username that is not the one used for installation.

5.1 UNDERSTANDING ENVIRONMENTS

The VLTSW is based on the concept of environments where processes can run and exchange messages with other processes. The communicating processes may be located in the same or in different environments, on the same host or on different machines.

Environment can be:

CCS-Lite (a.k.a. QSEMU) environment (on WS)

an environment providing database and communication facilities and used to build WS applications. On each WS there can be one or more WS environments.

LCC environment (on LCU)

provides database and communication facilities (lcu-Qserver) to build real-time applications. Maximum one environment per LCU.

An environment is uniquely identified by the environment name. Remember that environment names are limited to 7 chars and the first letter is mandatory “w” for WS and “l” for LCU.

REMARK: Real VLT environment names must follow the conventions defined by the applicable version of"VLT LAN's Specification". In the following example generic names built using the machine node name are used.

From the communication point of view, each environment is identified by the node on which is running and a TCP/IP port number. The same number can be used on different nodes for the same type of environment. Currently we use:

2160 for LCC environments,

one number in the range 2001-2999 for each CCSLite environment.

5.1.1 Configuring Environments

To be able to communicate, environments need to know about each other.

1. there is a TCP/IP channel (environment name and port number) for each environment. Actually, the couple node-number shall be unique in the system. So the same number can be used for all the LCC environments. TCP/IP channels are defined in the file `/etc/services`.
2. each CCS-Lite (QSEMU) environment needs to know where other QSEMU or LCC environments are located, they can be both local or remote to the WS. `$VLTDATA/config/CcsEnvList` provides such a mapping
3. each LCU needs to know from which host it has to boot and which are the other environments: CCSs and QSEMUs (an LCU should not talk directly to other LCUs). This is done by the `$VLTDATA/ENVIRONMENTS/<lcuenv>/bootScript` file. If LCU(s) need to communicate with more WS environments than the assigned boot environment, then the file "`$VLTDATA/config/lqs.boot`" must be created and edited. The standard file in "`$VLTROOT/vw/bin/$CPU/lqs.boot`" can be taken as template.

In addition:

each QSEMU environment shall have a `$VLTDATA/ENVIRONMENTS/<wsenv>` for the database files (`dbl`/ and `DB`/), snapshots and database image and process configuration table (CcsEnvTable)

each LCC environment shall have a `$VLTDATA/ENVIRONMENTS/<lcuenv>` for the database files (dbl/ DB/) and boot files (bootScript, devicesFile, ...)

In addition to the basic environment configuration, other configuration tables are required by applications like CCS Log System and CCS Scan System.

The verification process will drive you to a first set up of these files.

REMARKS: the VLT Software Environments Common Configuration User Manual provides a detailed documentation about environments and configuration tools. The panels layout as well as several examples of expected output are also given.

5.2 THE APPLICATION EXAMPLE

To have the possibility to work-out the CCSLite and LCC software, a simple example, consisting of an LCU application and of a WS application, is provided.

You can download the application examples from the SVN repository:

```
svn co $SVNREPO/trunk/VLTSW/Core/examples
```

Each application is implemented as a VLT software module and also provides an example of use of the VLT Programming Standards. Please note:

- the standard subdirectory structure,
- the use of naming conventions,
- the standard Makefile,
- the documentation in form of man-pages,
- the command definition table (LCU only),
- the error files.

5.2.1 The LCU Application

The LCU application consists of one process (`lcuapp`) able to treat the following incoming commands:

SETVAL with one integer parameter (ASCII format):

IF 0 < value < 100

THEN

- the value is written in the `:PARAMS:SCALARS.scalar_int32` point of the LCU local database
- the message “`Value received = <value>`” is logged
- an empty reply is given back

ELSE

- the message “`Value received = <value> (out of range)`” is logged
- an error message “`Value out of range`” reply is given back

EXIT:

- an empty reply is given back
- the process quits

The implementation of the LCU application is in `examples/lcuapp` and is generated during the installation. If needed, can be generated as follows:

```
$ cd ~/examples/lcuapp/src
$ make clean
$ make all
$ make man
$ make install
```

5.2.2 The WS Application

The WS application consists of two programs:

`wsappShowValue` that continuously displays an item from the local database until a “q” character is typed in.

`wsappSetValue` that works with a companion partner, named `lcuapp`, on another environment. It prompts the user for input:

IF an integer number is typed in:

- the number is formatted into a message (SETVAL) and sent to the partner
- the reply message from the partner closes the loop and prompts for a new input.

IF a “q” character is typed in

- an EXIT message is sent to the partner
- as the reply message from the partner comes back the process terminates as well.

The implementation of the WS application is in `/examples/wsapp` and is generated during the installation. If needed, can be generated as follows:

```
$ cd examples/wsapp/src
$ make clean
$ make all
$ make man
$ make install
```

5.3 INSTALLING AND CONFIGURING ACC DATABASE

The ACC database contains all the information (IP address, machine type, node name, etc.) used by the configuration tools and it uses `mSQL` as database engine.

This installation is done as "`vltmgr`". By default, "`vltmgr`" is also the username authorized to write the database. For more information, please refer to the ACC and msql documentation.

Only one database is required for all machines. The environment variable ACC_HOST tells the VLT Software the host where the database daemon is running. For every machine, including the one where the daemon runs, the following entry must exist in PECS:

```
export ACC_HOST=<host>
```

Please check the existence in /etc/pecs/releases/000/etc/locality/apps-`hostname`.env and install the mSQL software on that <host> only.

The mSQL code is generated and installed in `$TCLTK_ROOT`. Such a generation is sufficient to correctly generate the Sequencer. The following actions are required to create and start the database and to load it with the configuration data:

1. create user access list
   ```
   $ cp $TCLTK_ROOT/include/msql.acl $TCLTK_ROOT
   ```

(There should be not need to edit this file manually. Applying any changes while the daemon is running (see below) a subsequent call to the utility msqladmin reload is required to activate these changes.)

2. start the mSQL daemon:
   ```
   $  msqld&
   Mini SQL Version 2.0.11 ...

    (press enter to get the prompt back)
   ```

3. create acc database:
   ```
   $ msqladmin create acc
   Database "acc" created.
   ```

4. load the definition of the basic tables used by the VLT Configuration tools:
   ```
   $ msql acc < $VLTROOT/config/accCreateVCCTables.sql
   Welcome to the miniSQL monitor.  Type \h for help.
   mSQL >     ->     ->     ->
   ```

```
Query OK.

.......

Bye!
```

5. check database structure:

```
$ msqlrelshow acc
Database = acc

  +----------------------+
  |       Table          |
  +----------------------+
  | prog_environment     |
  | station              |
  | subnet               |
  | lcu_progenv          |
  +----------------------+
$ msqlrelshow acc station

Database = acc

Table    = station

+-----------------+----------+--------+----------+--------
-----+
|     Field       |   Type   | Length | Not Null | Unique
Index|
+-----------------+----------+--------+----------+--------
-----+
| station_name     | char       | 30       | N           | N/A
|
| station_type     | char       | 30       | N           | N/A
|
| subnet_name      | char       | 30       | N           | N/A
|
| ip_address       | char       | 20       | N           | N/A
|
| active_status    | char       | 15       | N           | N/A
|
| location          | char      | 80       | N           | N/A
|
| responsible       | char      | 80       | N           | N/A
|
| telephone         | char      | 80       | N           | N/A
|
| backplane_name   | char      | 40       | N           | N/A
|
| station_arch      | char      | 16       | N           | N/A
|
| station_cpu       | char      | 16       | N           | N/A
|
| station_bsp       | char      | 30       | N           | N/A
|
| attic             | int       | 4        | N           | N/A
|
| idx                 | index      | N/A      | N/A          | Y
|
+-----------------+----------+--------+----------+--------
-----+

.......
```

6. Create a plain ASCII data file. First get the template if not yet available under msql directory

```
$ cp $VLTROOT/config/accData.sql $VLTDATA/msql/
```

then edit the file `$VLTDATA/msql/accData.sql` as explained by the comments. When ready load it:

```
$ accLoadData
```

7. check the database content:

```
$ msql acc
Welcome to the miniSQL monitor.  Type \h for help.

mSQL > select * from station\g
Query OK.

.....

mSQL > \q
```

```
          Bye!
```

Test the installation by quering the database from a VLT utility:

```
          $ vccShow
          --- VCC: VLT Common Configuration ---
          ... Hosts defined: ..........
          ... Environments defined: .....
          ........
           Env      Host     TCPport  IP address        Arch     CPU      BSP
          -------- -------- -------- --------------   -------- -------- --
          -----
          l...
```

Remember to include the definition of ACC_HOST among the environment variables defined at the login.
The utility vccFastShow is also available, for big databases is much quicker.

Note: an example of the file accData.sql can be found on the web:
https://www.eso.org/sdd/bin/view/SDDPublic/VLTCommonSoftwareMain under the paragraph:
“VLT Common Software: templates to help the installation of a VLTSW machine”.

5.3.1 Automatic Startup of Database Daemon at boot Time

During the DVD installation, all the necessary links are generated under /etc/rc*.to have the database daemon always active after a reboot of the machine.

```
$ find /etc/rc[0-6].d/ -name "*msql*" -exec ls -al {} \;
lrwxrwxrwx  1  root  root  14  Oct    9  07:03  /etc/rc0.d/K09msql  ->
../init.d/msql
lrwxrwxrwx  1  root  root  14  Oct    9  07:03  /etc/rc1.d/K09msql  ->
../init.d/msql
lrwxrwxrwx  1  root  root  14  Oct    9  07:03  /etc/rc2.d/S91msql  ->
../init.d/msql
lrwxrwxrwx  1  root  root  14  Oct    9  07:03  /etc/rc3.d/S91msql  ->
../init.d/msql
lrwxrwxrwx  1  root  root  14  Oct    9  07:03  /etc/rc4.d/S91msql  ->
../init.d/msql
lrwxrwxrwx  1  root  root  14  Oct    9  07:03  /etc/rc5.d/S91msql  ->
../init.d/msql
lrwxrwxrwx  1  root  root  14  Oct    9  07:03  /etc/rc6.d/K09msql  ->
../init.d/msql
```

5.4 CONFIGURE AND VERIFY

This section guides you through the configuration of one CCSLite environment and one LCU. It is assumed that both VxWorks and CCSlite are installed.

If VxWorks is not installed, please skip everything concerning LCU.

5.4.1 Configure the LCU TCP/IP Channels

In order to have communication between LCU and WS (Engineering User Interface), a TCP/IP channels (environment name and port number) is required for each environment. TCP/IP Channels are defined in the file `/etc/services` (you need to be root to edit it). Add to such a file one line for each LCU and for each WS environment. E.g.:`/etc/services`

```
            .
            .
    #------ begin VLT configuration
    lted          2160/tcp        # lqs on ted
    wte19         2301/tcp        # ws environment on te19
    #------ end VLT configuration
```

5.4.2 Configure the LOGGING System.

The Logging System is not automatically started with the environment because it is independent from it. The Logging system is installed as follows:

1. log file location:
```
export VLT_LOG_FILES=/vltdata/tmp
export VLT_LOG_ROOT=/vltdata/config
```

These variables are normally set by the standard environment (alias PECS).

The following step 2, 3 and 4 are done during the automatic installation.

2. as vlt, create the empty files:
```
$ cd $VLTDATA/tmp
$ touch logFile
$ touch logAuto
$ chmod 666 logFile logAuto
```

   Be careful that the ownership of the files logFile and logAuto should be: vlt:vlt with permissions 666.

3. as root:
   a. rsyslog configuration:
```
vi /etc/rsyslog.conf
```

. . . . .

Check if the following entries have been correctly inserted by the installation procedure:

```
*.info;mail,local1,local2,local3.none;news.none;authpriv.none;cron.none         /var/log/messages


# The following two lines configure the VLT logging system
#
local1.warning<TAB>/vltdata/tmp/logFile
local2.warning<TAB>/vltdata/tmp/logAuto
```

Remark: All entries consist of one line each, with the key (first column) and its content (second column).

b. enable logging from remote hosts :

Edit `/etc/sysconfig/rsyslog` so that the SYSLOGD_OPTIONS line reads: SYSLOGD_OPTIONS="-c 5"

c. force `rsyslogd` to read the new configuration file

```
$ kill -HUP 'cat /var/run/syslogd.pid '
```

d. disable SElinux :

Edit `/etc/selinux/config` and check if the following line is present:

SELINUX=disabled

4. as `vlt`:

a. add the automatic clean-up of the log files to "vlt" crontab:

```
$ export EDITOR=vi
$ crontab -e
```

Add the following line (or substitute any previous line containing the `logVLTBackup` command)[1]:

```
0,5,10,15,20,25,30,35,40,45,50,55 * * * * bash -c \
'. $HOME/.bashrc; logVLTBackup vlt vlt 12 1000000'
```

5. as vlt, verify the syslog:

a. in one window: `$ logger -p local1.warning "this is my text"`

b. in another window: `$ tail -10 $VLT_LOG_FILES/logFile`

.... the second window should show your message.

[1]The last value of the line, "1000000", is the size that the logFile can reach before being backed up. This value can be reduced or increased according to the type of operations.

5.4.3 Data Flow System (DFS) logging system configuration (if applicable)

In order for the DFS logging system to keep track of VCSOLAC behaviour/performance, another entry must be added in the `/etc/rsyslog.conf` file on all the instrument workstations involved: this entry will send "local3.debug" messages to the OLAS machine (for Paranal's UTs a machine in the group wu{1,2,3,4}dhs).
For example add the following line:

```
local3.debug        @wu2dhs
```

in the /etc/rsyslog.conf file on the UT2 instrument workstations to send messages to the OLAS machine wu2dhs.

5.4.4 Configure the WS Environment

Each CCSlite environment is defined by a name that shall be unique in the network. Hereafter `<wsenv>` is used to name the CCSlite (or QSEMU) environment you are configuring. Substitute each occurency of <wsenv> with the real name.

1. as vltmgr, define the environment as the "local" environment, i.e., the one to access data from when environment name is not specified:

   ```
   $ export RTAPENV=<wsenv>
   ```

   Do it for all windows you may have opened. This definition is needed from now on and it should be added to the other ones defining the environment (e.g., added to `/etc/pecs/releases/000/etc/locality/apps-'hostname'.env`)[2]

2. make this environment known to the system:

   CCSlite

   Add a line as follows to `$VLTDATA/config/CcsEnvList`

   ```
    <wsenv>  <hostname>  <VLTDATA>/ENVIRONMENTS/<wsenv>      #
   environment on <ws>
   ```

   Starting with JAN2006, to support WS with multiple LAN interfaces corresponding to different hosts, the format of the CcsEnvList has been slightly modified. This new format of the CcsEnvList is fully supported by vccmake.

3. using the configuration tool:

   ```
   $ vccEnv &
   ```

[2]The standard environment provides the default definition: RTAPENV=w${HOST}

- enter the actual environment name and press Return (or, depending on the keyboard, Enter). Fields are filled with defaults values, normally appropriate.
- click "Create" button. The window will show the log of ongoing operation.
- click "Init" button. say "Continue" to confirmation window.
- click "Start" button. The logfiles in `$VLTDATA/ENVIRONMENTS/<wsenv>/*.log`
- or `$VLTDATA/ENVIRONMENTS/<wsenv>/.*.log` (note the initial "dot") should not report any errors.

The default database is already configured to support the verification tests. The CcsEnvTable file defines the processes that shall automatically start at database startup.

You can use the some CCS tools to inspect the database structure or to monitor the running processes (like ccsPerfMon).

5.4.5 Verify the WS Environment

The easiest way is to use ccsei to interact with the WS environment. ccsei is the interactive utility to exchange messages with local and remote applications, to read/write in databases, to monitor the logs, to call other development and debugging tools. The same functions are available for both LCU and WS environments.

If the WS environment has been properly started, all ccsei utilities should work. Invoke the main menu first:

```
$ ccsei
```

... the list of available tools is presented.

REMARK: from now on, the main menu of ccsei is supposed to be always displayed on the user screen.

Try some simple operations:

See a log:

1. from the ccsei main menu, click "CCS Log Monitor". The "VLT Log Monitor" tool is started, with by default the "MONITOR" option. On another xterm, create a log:

   ```
   $ logger -p local1.warning "this text should be displayed"
   ```

   It should be displayed.

Send a Message

2. click "CCS Command Window". The "ccsei Message Command Window" is presented
3. write "cmdManager" in the "Process" field, click "Help" and select "On Commands". The "Help On Commands" panel is presented.

4. <double-click> on the "VERSION" command: the command format is displayed and the command name also appears after the prompt of the Command Window (`1<wsenv>(Pro)> VERSION`).
5. click after the (Pro) > `VERSION` and press the <Enter> key to send such a command to the cmdManager process. The "Replies" field should give the version number of the currently running `cmdManager`.
6. close the "Help On Commands" window;
7. click "File" and then "Quit" to close the "ccsei Message Command Window"

Read/Write into the database:

8. click "CCS Database Monitor" in the main menu. The "ccsei Database Monitor" window is presented. Click "CCS Database Monitor" again to have a second window. The first will be used for a continuos monitor of a variable, the second to change the variable content.
9. in both windows: double-click on "PARAMS", "SCALARS", then select "scalar_int32". Both should display the same value.
10. In the first window, click "Move to list", double-click on "scalar_int32" and "Activate Monitor"
11. In the second window, click on the "DB Value" field, type another value and press <Apply>. The first window should show the new value in the field labeled "Values:" (a little delay is normal).
12. close all the panels, but the main menu.

More about the ccsei can be found in the "CCS User Manual".

5.4.6 Configure the LCU Files on the WS

To work with the LCC requires that some files are set on the WS acting as boot node.

Each LCU environment is defined by a name that shall be unique in the network. Hereafter:

`<lcuenv>` is used to name the LCU environment you are configuring

`<lcunode>` is the LCU node name.

Substitute each occurency of `<lcuenv>` and `<lcunode>` as appropriate.

For each LCU environment:

1. be sure that:
    - the LCU is booted and the remote login from the WS to the LCU is possible (no one is locking the LCU shell).
    - the "vx" username is defined and the LCU can execute a remote shell to it (`~vx/.rhosts` correctly set)
    - the ownership of the file `~vx/.rhosts` is : vx:vlt with pemissions 644.
2. using the configuration tool:

```
$ vccEnv &
```

- enter the actual LCU environment name (<lcuenv>) and press Return (or, depending on the keyboard, Enter). LCU-host (<lcunode>), the Boot-env (<<wsenv> or <qsemuenv>) are taken from the database.
- click “Maximum” and then click on “Create”.
- click “Init”.
- the “Config...” button invokes the “vccConfigLCU” panel to change configuration options. The default values are enough for the validation test, but you can set what needed by your applications. Click “WriteFiles” when ready to regenerate target files. Click “Continue” to override the current files. Then go back to the vccEnv panel.
- click “Start” to reboot the LCU. The log of the reboot process allows you to follow it. At the end of the boot, the console should display the message:

```
LCC INITIALISATION SUCCESSFUL.
```

REMARK: to be able to display the boot log, the LCU Configuration tool locks the connection to the LCU for about 120 seconds. After this time the connection is released (`vccConfigLcu: connection closed`). If the boot process takes longer, part of the log may not be displayed. In such a case, check the LCU console.

3. check that among running LCC processes there are:

```
$ rlogin <lcunode>

<lcuenv>->i
    (the processes may be listed in different order!)
  NAME         ENTRY        TID     PRI   STATUS    ......
---------- ------------ -------- --- ---------- ......
  ......       ....         .....
  ......       ....         .....
lccServer  cmdInit        61d964  80 PEND+T
rdbServer  cmdInit        6005cc  80 PEND+T
msgServer  msgServer      6864f8 100 PEND
lccTime    737fa0         681548 100 DELAY
lccEvent   71848a         652438 100 READY
lccPeriodic733a92         64d488 100 READY
lccDevice  711d1e         622914 100 DELAY
lccWatchdog73d86e         6c7e50 150 READY
tLqsClk    784410         6ce1bc 250 DELAY
lccLogger  728d18         68b4a8 250 READY
value = 0 = 0x0
```

```
<lcuenv>->
```

and that the LCU environment table has been correctly loaded

```
<lcuenv>-> lqsPrintEnvTbl
Environment   Host name   TCP port   .....  .....
-----------   ---------   --------
  wte19         te19         2226
  lted          ted          2160
value = ... (some numbers here...)
<lcuenv>->
```

In case the LCU does not boot correctly check that: the boot directory is NFS mounted:

```
<lcuenv>-> nfsDevShow
device name          file system
-----------          -----------
/vltdata             wtatcam:/vltdata
/VLTROOT             wtatcam:/vlt/OCT99
value = 0 = 0x0
<lcuenv>->
```

5.4.7 Make the LCU known to the CCSLite environment

1. make the LCU known to CCS: add a line as:

   ```
   <lcuenv>  <lcunode>   # LCC environment on <lcu>
   ```

   to:

   ```
   CCSlite: $VLTDATA/config/CcsEnvList
   ```

2. configure reporting node for LCU log activity. You should copy a template file from the $VLTROOT under your WS environment directory. Execute the following:

   ```
   $ cp $VLTROOT/include/logLCU.config.template \
               $VLTDATA/ENVIRONMENTS/<wsenv>/logLCU.config
   ```

and modify it according to your configuration.

```
$ cat $VLTDATA/ENVIRONMENTS/<env>/logLCU.config
<lcuenv> <wsenv>       (*)
```

(*) remember to terminate the last line with EOL

See also: man logCreateLcuConfig

3. check the connection
    a. by a simple message exchange:

```
$ msgSend <lcuenv> lccServer LCCGVER ""
  MESSAGEBUFFER:
  "@(#) LCC ....<version>... ....<date>....."
```

4. force the logManager to read the new configuration:

```
$ logConfig
```

5. from the `ccsei` main menu, start a "CCS Log Monitor", click on "MONITOR", then create a log from the LCU

```
$ rlogin <lcu>

<lcuenv>-> logData "test", 101, "this is a log from an LCU"
value = 2 = 0x2
```

   check that the log is displayed by the monitor tool.

   REMARK: a delay of few seconds is normal because, in order not to overload the network, logs are transmitted by the LCU to the upper level only periodically.

5.4.8 Verify the LCU environment

`ccsei` can also be used to verify the just created LCU environment.

Send a Message

1. click "CCS Command Window". The "ccsei Message Command" window is presented
2. from the [+] menus, select `<lcuenv>` as environment and then `lccServer` as process.
3. send the command `LCCGVER` (either by typing it after the prompt or by selecting from the menu); type 'enter". The "Replies" field should display the version number:

```
REPLY: "@(#) LCC ....<version>... ...<date>..."
```

   In the same way, other commands can be used (see LCC User Manual)

Read/Write into the LCU database:

4. click "CCS Database Monitor" in the main menu. The "ccsei Database Monitor" window is presented. Click "CCS Database Monitor" again to have a second window. The first will be used for a continuos monitor of a variable, the second to change the variable content.
5. continue as in Verify the WS, but from the [+] menus, select `<lcuenv>` as environment . Notice that the selection of the database point using the "Browse" facility is slower because the system queries the database structure from the running LCU.

5.4.9 Interact with an LCU Application

The lcuapp is used to test and demonstrate the way of working of `ccsei`.

1. from the ccsei main menu, click "Log Monitor". The "VLT Log Monitor" tool is started. Enable the monitoring by clicking the "MONITOR" option.
2. make the `lcuapp` known to `ccsei`. Add one line containing `lcuapp` to the file `$VLTDATA/ENVIRONMENTS/<lcuenv>/PROCESSES`
3. load `lcuapp` into the LCU and start it: open an rlogin on the LCU. Give the following commands

```
$ rlogin <lcunode>

<lcuenv>-> cd getenv("VLTROOT")
<lcuenv>-> cd "vw/bin/MC68040" (if you use 68k LCUs)
or
<lcuenv>-> cd "vw/bin/PPC604"  (if you use PPC LCUs)
<lcuenv>-> ld < lcuapp
<lcuenv>-> sp lcuapp
```

   Two logs should appear in the LCC Log/Inspect Window telling that `lcuapp` has started and is waiting for commands.
4. start a "CCS Database Monitor" window in monitoring mode (check "Move to list" and "Activate monitor") on the <lcuenv>`:PARAMS:SCALARS.scalar_int32` variable.
5. from a "CCS Command Window", select `<lcuenv>` as environment and then `lcuapp` as process, then send a `SETVAL <nn>` message to `lcuapp` to change the value in the LCC database. <nn> is an integer; the valid range is 0-100.
   - the Database Monitor should show the new value (`<nn>`)
   - the Log Monitor should report on the LCU activities.
   - the Command Window should not display error replies

   Repeat for different values. If `<nn>` is out of range, an error and a different log should be received.
6. from a "Command Window", send `EXIT` to `lcuapp` to stop the application.

5.4.10 Verify Cooperation between WS and LCU Applications

The scenario we are going to use now is the same as the one used to test the `lccei`, but a WS application is used to send commands to the LCU companion.

At this point you are already an expert in configuring both WS and LCU so set things up as follows:

- the WS environment is running (see 5.4.4)
- the Logging system is active (`logManager)` and a `logMonitor` window is displayed (see 5.4.2, 5.4.7)

- th LCU is properly configured, i.e., it can communicate with the WS environment (see 5.4.7)
- `lcuapp` is running on the LCU (see 5.4.8)
- `ccsei` is monitoring the LCC database position `:PARAMS:SCALARS.scalar_int32` (see 5.4.9)

At this point you can run the WS application that sends the number you type to `lcuapp` in order to be stored, if in the range, into the LCC database.

```
$ wsappSetValue <lcuenv>
wsappSetValue - WS application example - Version 1.5, June 94

Enter a number to send to the LCU ....
Enter 'q' to quit
<nn>
<nn>
....
q
```

If in range, the number you type will be displayed by the Monitor window. Repeating for different values of `<nn>` you can experiment with several cases, including logs, errors, replies, etc. Type a "q" to stop both `lcuapp` and `wsappSetValue`.

These actions complete the verification of CCS/CCSLite and LCC installation.

5.4.11 WS Environment Shutdown

To terminate the verification, close the environment properly using the vccEnv "Stop" or:

```
$ vccEnvStop -e<wsenv>
```

These actions complete the verification of CCSLite and LCC installation.

5.5 SUMMARY: LIST OF FILES NEEDED FOR THE CONFIGURATION OF ENVIRONMENTS (LCU AND WS)

For WS environments:

- **$VLTDATA/msql/accData.sql**: update the tables "station" and "prog_environment" (see also template on the web at http://www.eso.org/projects/vlt/sw-dev/vlttemplate/accData.sql);
- Please remember to apply the command `accLoadData` after editing this file.
- **/etc/services**: write a couple "wsenvname port_number/tcp"; by default port numbers should run in the range 2001-2999;

- **$VLTDATA/config/CcsEnvList**: write
  a. wte49 te49/vltdata/ENVIRONMENTS/wte49 (if the environment is local, wte49 is the name of the ws environment)
  b. wte35 te35 (if the environment is on a different workstation, wte35 is the name of the environment while te35 is the host the environment belongs to)

  Note: WS environments on different machines require a properly aligned setup to enable communication. Specifically the accounts running the environments must exist with same UID/GID on both machines, and both environments' names and port numbers must match.

For LCU environments:

- **$VLTDATA/msql/accData.sql**: update the tables "station", "prog_environment" and lcu_progenv (see also template on the web at `http://www.eso.org/projects/vlt/sw-dev/vlttemplate/accData.sql`);
- Please remember to apply the command `accLoadData` after editing this file.
- **/etc/services**: write a couple "lcuenvname port_number/tcp"; from a communication point of view, each environment is identified by the node on which is running and a TCP/IP port number. The same number can be used on different nodes for the same type of environment. Currently we use the value 2160 for LCU environments;
- **$VLTDATA/config/CcsEnvList**: write a couple "lcuenvname lcunodename" for every lcu environment that has to be known on that WS host;
- **$VLTDATA/ENVIRONMENTS/<wsenvname>/logLCU.config:** use the utility logCreateLcuConfig to prepare the file, which will contain the couple "lcuenvname wsenvname", specifying which LCUs have a given WS environment as logging reporting node;
- **~vx/.rhosts:** the file .rhosts of the user vx must contain all LCU nodes (hostnames) that have to work with the system (vx must be able to open a remote shell on those hosts). The permission of the file should be 644 (command: `chmod 644 ~vx/.rhosts`).

The tool vccmake can be run as vltmgr to update the following files from the configuration database:

- $VLTDATA/config/CcsEnvList
- /etc/services
- ~vx/.rhosts

5.6 OTHER TESTS

The verification procedure has the purpose to check that basic features work correctly and to get you acquainted with the VLT Software.

During an installation you may have decided to install more modules than tested in the verification procedure. The User Manual of each software module contains additional tests. Please refer to such documentation to configure and test drivers, INS software, etc. as appropriate to your installation.

5.7 CONFIGURING WORKSTATION STARTUP

As appropriate, the startup of the standard environment, qsemu, logManager, etc. can be part of the WS startup process. The automatic startup of the ACC database server is described in Automatic Startup of Database Daemon at boot Time, the automatic start of the WS environment is described in the CCSlite installation procedure.

5.8 REPORT TO ESO

You are kindly requested to provide by mail the list of the products you have installed, including the computer configuration (type, OS, etc).

Please contribute the information requested above to improve the quality of this manual, especially the trouble-shooting list (see 6). Add to your report any suggestions you may have to improve the installation procedure or any mistakes you may have discovered.

Problems or errors in the installation procedure should be reported using the JIRA tool (https://jira.eso.org) (see 3.6).

6. TROUBLE SHOOTING GUIDE

This section is intended to provide a first level debugging help. More can be found in the specific documentation. The following list is, of course, not exhaustive, and consists of cases having occurred during previous installations. Please feel free to contribute your experiences (see 5.8). Information about a case you have encountered and solved which is not described in the following section may prove useful for somebody else. Let's help each other to save time!

6.1 Problems with NFS files

NFS does not permit mounting a disk already mounted with NFS. Check that each NFS mount is to the machine where the files are really located.
When symbolic links are used the full path must be given (e.g. `/diska/vltdata` instead of `/vltdata`, when `/vltdata` is a symbolic link to `/diska/vltdata`).

6.2 The LCU does not boot

- check that the LCU is properly connected to the network
- on the LCU check the boot parameters (see 5.4.6): the host name, where to look for the VW boot file, the username
- on the WS check that: the "`vx`" username is defined and that the VxWorks file is readable by such a user.
- check that "vx" doesn't have a .cshrc that produces output!

6.3 The LCU boots but does not find the bootScript file

- on the LCU check that the bootScript file is correctly specified. Remember that NFS does not manage links: you shall specify the physical absolute path.
- on the WS check that:
  - VLTROOT can be NFS mounted
  - the file is readable by the "vx" user (see 5.4.6)

6.4 LCC does not start successfully

- on the WS check that:
  - VLTROOT can be NFS mounted
  - bootScript is correct and contains the absolute path to VLTROOT and BOOTROOT
  - the user specified in the bootScript exists on your UNIX system. To make it simple, use "vx" for both booting the LCU and in the bootScript file.
  - logfile and rebootFile are available in $BOOTROOT/ENVIRONMENTS/<lcuenv>, even as empty files, and readable/writable by the "vx" user.

- devicesFile is available in the $BOOTROOT/ENVIRONMENTS/<lcuenv>. Remember that even if you do not have any devices, the file must exist and with a minimal content, see devicesFile(5) and being readable by the "vx" user.

6.5 VxWorks code is not compiled

- check VxWorks environment (see 4.3.3)

6.6 WS environment does not start

- check CCS environment (see 5.4.4)
- check that operating system parameters have been changed according to the CCSlite installation.
- if the logfile: $VLTDATA/ENVIRONMENTS/<env_name>/.ccsScheduler.log reports the following

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
ccsScheduler: S-BIT not set for ccsScheduler1
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

run the following commands:

```
chown root /vlt/<RELEASE>/CCSLite/bin/ccsScheduler1
chmod u+s /vlt/<RELEASE>/CCSLite/bin/ccsScheduler1
```

6.7 Lccei does not start (properly)

- check that the `PATH` contains `$VLTROOT/bin`
- check that `$VLTROOT/config/Lccei` has been loaded by `xrdb`
- check that `$VLTROOT/CDT` contains valid files (the format can be checked using `lcccdt`), one for every process defined in `$VLTDATA/ENVIRONMENTS/<lcuenv>/PROCESSES` file
- check that qsemu is running

6.8 Lccei cannot communicate with an LCU

- check that `qsemu` is up and running
- check `$VLTDATA/config/`logLCU.config
- check that the environments defined on WS (`qsemu -env`) and LCU (`lqsPrintEnvTbl`) are consistent.

6.9 LOG system does not work

There are several facts that may affect the syslog, either at system level or due to bad VLT Software configuration. The following may help to find some of the possible causes:

- check that `syslog` is working:

```
        $ ps -efa | grep syslog
```

- create a log in one of the system files (the commands used in the following example are for documentation only and may not work in your configuration. Please check them with your system manager):
  - have a look in `/etc/syslog.conf` and chose a "selector"
  - use logger to create an entry in the corresponding log file :
    ```
    $ logger -p mail.debug "aaa"
    ```
  - check that the file associated with the selector
    ```
    $ tail /usr/spool/mqueue/syslog
    Mar 06 17:14:44 te13 vltsccm: aaa
    ```
- check that the VLT_LOG_... variables are defined and that the content of `/etc/syslog.conf` is consistent with such a definition. Restart the syslogd to be sure that it is using the right files:
  ```
  $ kill -HUP 'cat /etc/syslog.pid'
  ```
- put a log in each of the log files:
  ```
  $ logger -p local1.warning  "this should go in logFile"
  $ tail $VLTDATA/tmp/logFile
  ................
  $ logger -p local2.warning  "this should go in logAuto"
  $ tail $VLTDATA/tmp/logAuto
  ................
  ```
- if you have CCS environment, create a log from the WS :
  ```
  $ logUserData aa aa aa
  $ tail $VLTDATA/tmp/logFile
  ................
  ```
- if you have an LCU, create a log from the LCU:
  - check that `$VLTDATA/config/logLCU.config` exists as well as the pair `<lcuenv><wsenv>`. If needed edit the file.
  - check that `logManager` is active. If not start it: `$ logManager &`
  - force logManager to read the configuration file: `$ logConfig`
  - on the LCU, give the following command to create a log
    ```
    $ rlogin <lcu>
    <lcuenv>-> logData "test", 101, "this is a log from an LCU"
    ```

Remember that `logMonitor` can be used to display the current VLT logs.

6.10 lccei does not accept CDT

CDT in NOV95 format or older are not valid any longer. Please edit them according to the new format as described in the LCC User Manual.

6.11 Scan system does not start on the LCU

LCU Database in FEB95 format or older are not valid any longer. Please change the LCU DB according to the new format.

6.12 Declarative conflicts during installation

The include files of this release cannot be mixed with previous ones. May be you have forgotten to save the old installation before starting this installation (see 4.2.2).

6.13 Security error messages from Tk

Q.C4- X server insecure (must use xauth-style authorization) Tk requires you to have a secure X server before you can use the send command.

See the question 2.A.7 “How can I get Tk3.3 to even start - I get security error messages.” in Thomas Accardo’s Tk Toolkit Usage FAQ as well as `http://ce-toolkit.crd.ge.com/tkxauth/` for instructions on how to make your server secure.

6.14 LCU boot problem: INTROOT contains old code

You may experience problem at boot time if you have INTROOT defined and old software is loaded there.

`lcuboot’ will load modules automatically from INTROOT if they are there. If, for any reasons, old versions of LCC of of some other modules loaded during the boot are stored there, the boot process may be not complete or, if severe errors - like unresolved symbols - are found it may also abort.

After installation of any new version, current INTROOTs, if any, should be cleaned and the software regenerated using the newest version in VLTROOT.

6.15 LCU boot problem: tim board not installed

When LCC tries to read the time-board. If no board is installed, an error message is produced but the booting process continues regularly. If you do not have such a card, please ignore the message.

6.16 “Unbalanced” Parentheses Warning from tclCheck

The “warnings” arise from the utility `tclCheck’, which is automatically called by vltMake for every TCL file. Such “unbalanced” parentheses warning in correct code can be removed by using “\” as in the following examples:

```
> > File ccseiDb.tcl:
> > Inside a string: unmatched ( ending line 294 char 27
> >
> 294:             set i [string first "(" $att]
E.g.: replace by: set i [string first "\(" $att]

> > File ccseiDbBrowser.tcl:
```

```
> > Expecting } got ] : line 309 char 31
> >
> 309:    # -attrGeometry $attrGeometry]
also:     # -attrGeometry $attrGeometry\]

Line 385:      set line " ("
replace with: set line " \("

Line 415:      append line " 1==1)"
replace with: append line " 1==1\)"
```

6.17 Different LCU environments on the same node

If there are several environments assigned to the same node and, by mistake, one writes two or more of them in the logLCU.config file, sometime errors occur (no reply from LCU, message “environment not active” etc.).

NEVER PLACE IN logLCU.config two or more LCU environments associated to the same node!!!

6.18 LCU slow speed.

During ASM commissioning, we encountered a dramatically low speed of the operations on a LCU: A process supposed to complete in 16 seconds was running more than 5 minutes. The investigation revealed that a flag had been set on the LCU, enabling run-time printouts on the LCU console, while no rlogin <lcu> had been issued. The effect is that the serial line (9600 bps) is used to issue the approx. 300000 characters output during the sequence. A rough calculation of the output time is:

300000 char * 10 bits/char / 9600 bps > 300 seconds (= 5 minutes)

Therefore, take care of this kind of setting (the flag was set in the userScript) that may gravely impede the performance of an LCU if no rlogin is used.

A. PECS (Pluggable Environment Contribution System)

A.1 USER ENVIRONMENT SETTINGS

PECS supports personal settings for the following types of settings:

- Environment variables
- X resources
- GUI root window menu items
- shell aliases

All personal settings belong in files in the ~/.pecs directory, which should exist and contain (comments-only) example files. The big problem with putting a setting in one of these files, is knowing which file to put it in! This section explains the steps to determine where to put a setting:

1. Decide the ’type’ of the setting

Four ’types’ of personal settings are supported by PECS:

a.	environment variables	(type=env)
a.	X resources	(type=xrdb)
a.	GUI root menu items	(type=wmrc)
a.	shell aliases	(type=ali)

Note down what ’type’ it is.

2. Decide the ’application-scope’ of the setting

Settings either:

a. relate to a PECS-aware application (application-scope=apps)
   (e.g. VLTROOT, INTROOT, GNU_ROOT, NOCCS, PRINTER, RTAP_EXISTS)
b. don’t relate to a PECS-aware application (application-scope=misc)
   (e.g. DFLOW_ROOT because it relates to older versions of VLTSW only, NNTPSERVER, LESS, MAIL, PATH=$HOME/bin:$PATH).

3. Decide the ’host-scope’ of the setting

Settings either:

a. are applicable only on certain hosts (host-scope=<name-of-host>)
c. are applicable on all hosts (host-scope=all)

4. Put the setting in the right file

Now that you’ve worked out the three attributes ’type’, ’application-scope’ and ’host-scope’, it’s easy; the setting belongs in the file:

    ~/.pecs/<application-scope>-<host-scope>.<type>

e.g., to change your prompt set ’PS1’ in ~/.pecs/apps-all.env, and
to change your VLTROOT set VLTROOT in the same file.

5. Activating the setting

To activate the change in setting first run:

```
make_xdefaults
```

and then log out and back in.

Let’s look at a couple of examples; consider each of the following settings, look at the values it has for each attribute, and see the name of the file it goes in.

Example #1: Changing VLTROOT

Suppose you’re working on a machine where VLTROOT is set to /vlt/MAR2001/CCS, and you need to point it at FEB2000 instead just for a quick test. Well,

it’s an environment variable (type=env)
it relates to VLTSW, which is a PECS-aware application (application-scope=apps)
you only need it for a quick test on host ’te13’ (host-scope=te13)

So, having worked that out, the setting belongs in the file:

    ~/.pecs/apps-te13.env

So, in the file you just put:

    VLTROOT=/vlt/FEB2000/CCS

Note that environment variables supported by a PECS-aware application are automatically made available to sub-processes (’exported’) so you do not need to export it yourself (though it does no harm to do so).

Example #2: the ’which’ alias

Yes, the ’which’ command under Bourne-like shells does not work quite the same as it did under Tcsh. Under Bash, the nearest equivalent to ’which’ is ’type’ or ’type -all’. So to make ’which’ an alias for ’type -all’:

it’s a shell alias (type=ali)
it’s not related to a PECS-aware application, it’s just personal setting (application-scope=misc)
you want it everywhere (host-scope=all)

So, having worked that out, the setting belongs in the file:

```
~/.pecs/misc-all.ali
```

So, in the file you just put:

```
alias which='type -all'
```

Aliases are read per-shell-process, so it is not necessary to log out; your next new xterm will see the alias.

Note that Bash aliases have an '=' between the alias and its value, unlike Tcsh.

Example #3: you want to add 'top' and 'xv' to your 'User Menu'

PECS provides a GUI root window sub-menu entitled 'User Menu'. Suppose you want to create the menu to contain the command
'top'

it's a GUI root window menu item (type=wmrc)
it's not related to a PECS-aware application (the user menu never is) (application-scope=misc)
you want to see this menu everywhere (host-scope=all)

So, having worked that out, the setting belongs in the file:

```
~/.pecs/misc-all.wmrc
```

So, in the file you put:

```
Menu user_menu
{
  "My Menu"   f.title
  "Top"       f.exec "xterm -T top -n top -e top"
  "Xv"        f.exec "xv"
}
```

Note that 'top' isn't on all systems, so simply calling 'top' is a bit naive.

To activate the settings select 'Restart' from the GUI root window menu.

Example #4: you don't like the Xclock updating every second

PECS allows you to specify new values for X resources, but it also allows you to 'unset' existing ones (this is not a standard behaviour of X).

You will probably find your Xclocks showing a second hand, because of the X resource:

```
XClock*update: 1
```

To cancel this setting, just provide a blank value for it. So,

it's an X resource (type=xrdb)
it's related to a PECS-aware application (yes, this resource is defined by VLTSW) (application-scope=apps)
you want to see this menu everywhere (host-scope=all)

So, having worked that out, the setting belongs in the file:

```
~/.pecs/apps-all.xrdb
```

So, in the file you put:

```
XClock*update:
```

To activate the change you need to log out and log in again.

Example #5: you don't like to use "export VARIABLE=VALUE" and you want to keep using "setenv VARIABLE VALUE"
Very bad idea, but if you are so obstinate, you should write a function, because aliases which require embedded parameters must be written as functions, but should still be put in the ".ali" file. Here's an example:

```
export_function()
{
  export $1=$2
}
alias setenv=export_function
```

A.2 A CRASH-COURSE IN BOURNE SHELL

What follows is a little cookbook of common things you may wish to put in your ~/.pecs/*.env files:

To set a variable do this:

```
VARIABLE1=value
VARIABLE2="value with spaces"
export VARIABLE1 VARIABLE2
```

(Notice there are no spaces around the '='; bourne shell is sensitive about this.)

To safely test if a variable is set do this:

```
if [ "X$VARIABLE" != X ]; then

  code to do if VARIABLE is set

else
```

```
    code to do if VARIABLE is not set

fi
```

To set an alias, put something like this in your ~/.pecs/misc-all.ali file:

```
alias ALIASNAME="alias value"
(e.g: alias ls="/bin/ls -a")
```

A switch statement looks like this:

```
case "$VARIABLE" in

    value1) do this
            and then this
            and this is last ;;

    value2) do this
            and something else ;;

    *)      this is the default action ;;

esac
```

If you want to do a tcsh rehash, you should do:

```
hash -r
```

--- End of document ---