“CCS: Central Control Software”

- Layer of common services for applications
  - CCS on WS
  - LCC on LCU
  - Same API on both platforms.
- WS: built on top of RTAP
  - Industrial SCADA from HP
  - Based on recognized standards
  - Real time Database
- CCS isolates applications software from RTAP/Unix
- CCS implements the ESO specific conventions.
- CCS-Lite: Reduced functionality, without RTAP.

MODULES:

- CCS: Basic common functions.
- ERR: Error reporting.
- LOG: Logging system.
- MSG: Message system.
- CMD: Command checking.
- *DB: On Line Database.*
- *EVT: Event handling.*
- *TIMS: Time calculations & triggering.*
- NTP: Clock synchronisation.
- *SCAN: Data propagation.*
- ALRM: Alarm reporting.
- BOOK: Security checks.
- PANEL: GUI development support.
- CCSEI: Engineering UIF.
- SAMP: Sampling/Plotting tools.
- DBL: Database generation.
- ECCS: CCS C++ API & Classes.
- EVH: C++ Event Handler template.

Italic indicates modules partly available under CCS-Lite, red flags modules non avaialable under CCS-Lite.

ENVIRONMENT:

- originally RTAP Concept,
- – Scheduler (Rtap/Ccs)+
- – other core processes +
- – CCS applications +
- – user applications +
- – communication channels +
- – database +
- – other resources
- Only one per LCU.
- Single or multiple on WS.

- REMARK:In order to benefit from all the Scheduler facilities, all processes running in an environment must be declared in the corresponding Rtap(Ccs)EnvTable

EnvTable:

```
#   P       R  M  C  G    S   P    C
#   R       E  U  L  E    H   R    O
#   O  P  U S  L  E  T    U   I    M
#   C  H  S T  T  A  O    T   O    M
#   N  A  E A  I  N  B    D   R    A
#   U  S  R R  P  U  I    W   T    N
#___M__E__?__T__L__P__T_____N___Y_____D______________________________________________
```

These are the core processes which I will always want to run.

1 0 N N N N N 100 128 RtapScheduler

2 1 N P N N Y 99 128 RtapMonitor -a -s > rtap_log 2>&1

3 1 N P N A Y 100 128 RtapQServer

4 1 N N N A N 80 128 RtapEventTrig

5 1 N N N A Y 80 128 RtapEventConfg

6 1 N E N A Y 80 128 RtapTimeKeeper

First thing for CCS: setup of the shared memory area.

0 1 Y N N A N 100 128 ccsSHManager -s 20000

7 1 N N N A N 1 128 RtapDbStartup

#RtapDbStartup requests to proceed to next phase. i.e. phase 2

58 0 N E N A N 80 128 RtapMQDBM

9 2 N Y N A N 1 128 RtapDbCfServer

Miscellaneous Tasks which may have to be performed.

0 2 N P N N E 100 128 logManager

CCS tasks

10 2 Y N N A N 20 128 RtapScanMngr -m 100000

11 0 Y N N Y Y 1 128 ccsScan

0 0 Y N N Y Y 1 128 scanOut

0 0 N N N N N 1 128 scanConfig

0 2 Y E N A N 100 128 envsKill

13 2 N N N A N 100 128 msgServer

14 2 N P N N N 100 128 cmdManager

...

CCS Programmatic Library

The CCS library provides programmatic access to common resources such as:

- Connecting to an environment.
- File handling in the VLT file system structure.
- Allocating and releasing memory resources
- Definition of signal handlers for special commands

ccsInit	Initializes resources necessary for further access to CCS
ccsExit	Performs cleanup operations before exiting a process.
ccsFindFile	Searches for a file in the VLT file system hierarchy.
ccsGetMyEnvName	Returns the name of the environment a process is registered to
ccsGetMyProcId **ccsGetProcName** **ccsGetProcNum**	Returns process information such as process name, process Id and env. name.
ccsOpenFile	Open a file in the VLT file system
ccsGetProcNum	Returns the process number of a process
ccsWaitForProStat	Used to wait for a process to be in a given state: ready to accept command, to answer command, or already exited.
ccsNextPhase	Triggers the Scheduler to proceed with the next startup phase.

MESSAGE SYSTEM:

- IPC standard:
  - Unix queues within environment
  - Sockets between environment
- Commands/Reply(ies)
- Filtering on reception: any message, any reply, only replies from a given process, or for a given command, or for a given commandId
- CDT.
- Command checking.
- Parsing.
- Simulation.
- Booking checks.

Command Definition Tables

- different sections: Public, Maintenance, Test.
- Command + Synonyms.
- DATA Format: ASCII or Binary, Free format.
- Parameters description:
    - Name
    - Unit
    - Optional or Mandatory
    - Static range
    - Default value.
- Help text
- Simulation data for Reply. Possible DB Ref.
- Special commands: KILL/BREAK, PING, MSGE/DLOG

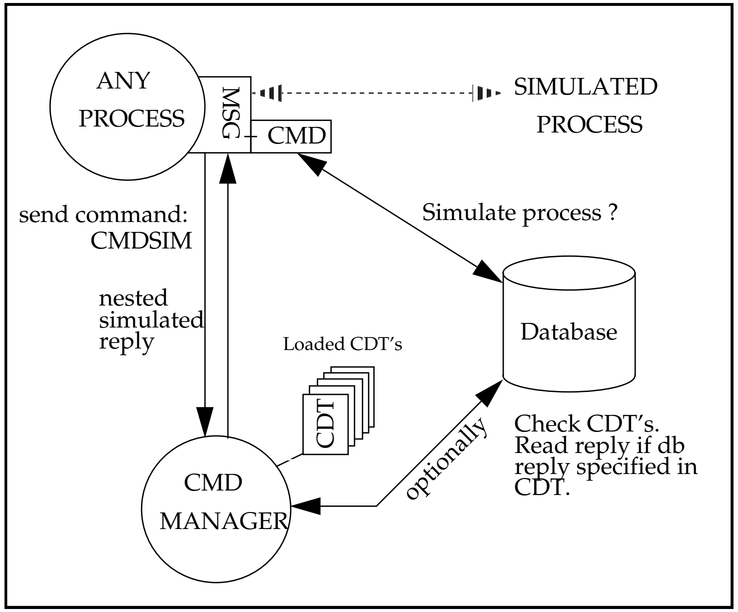
CDT Example:

```
COMMAND= MOVE
SYNONYMS=
FORMAT= A
PARAMETERS=
   PAR_NAME=   waveShape
   PAR_UNIT=
   PAR_TYPE=   STRING
   PAR_RANGE=  ENUM "SINE", "TRIANGLE", "SQUARE", "CONSP"
   PAR_DEF_VAL= "SINE"

   PAR_NAME=   amplitude
   PAR_UNIT=   deg
   PAR_TYPE=   REAL
   PAR_RANGE=  INTERVAL MIN =-1.5; MAX = +1.5
   PAR_DEF_VAL= 0.0

   PAR_NAME=   period
   PAR_UNIT=   sec
   PAR_TYPE=   INTEGER
   PAR_RANGE=
   PAR_DEF_VAL= 10

   PAR_NAME=   duration
   PAR_UNIT=   sec
   PAR_TYPE=   INTEGER
   PAR_RANGE=
   PAR_DEF_VAL= 10

REPLY_FORMAT= A   // "OK"

HELP_TEXT=
The axis will be moved, in position control mode, according to the specified
<waveShape>, <amplitude> in degrees, <period> in seconds (Only integer number
of seconds). For periodic functions, the motion is +/- amplitude around the
starting position
.If constant speed is selected, the axis is moved with <amplitude> degrees
over <period> seconds. Speed is thus <amplitude>/<period> degrees per second.
The motion starts from current position, and stops after <duration> seconds.
@
```

cmd interface:

cmdCheckCmd().	Check command parameters and inserts defaul values.
cmdParseCDT().	Loads a CDT file into the cmd system.
cmdParamList().	Parse message body to a list of parameters.
cmdParamFree().	Release allocated parameters list memory.
cmdParamDefault()	Build a parameter structure filled with defaults from CDT.
cmdParamDelete()	Delete parameters from parameter structure.
cmdParamGetByIndex().	Get values of parameter indexed by parameter number.
cmdParamGet<TYPE>ByIndex()	For each type of parameter an convenience function exists that does handle the type: STRING, INT, LOGICAL, REAL.
cmdParamGetByName().	Get values of parameter indexed by parameter name.
cmdParamGet<TYPE>ByName()	For each type of parameter an convenience function exists that does handle the type: STRING, INT, LOGICAL, REAL.
cmdParamSetByIndex().	Set values of parameter indexed by parameter number.
cmdParamSet<TYPE>ByIndex()	For each type of parameter an convenience function exists that does handle the type: STRING, INT, LOGICAL, REAL..
cmdParamSetByName().	Set values of parameter indexed by parameter name.
cmdParamSet<TYPE>ByName()	For each type of parameter an convenience function exists that does handle the types: STRING, INT, LOGICAL, REAL.
cmdParamToMsg().	Revert parameter list into message body.
cmdParamFree().	Relase internal memory allocated by cmdParamList.
cmdSimulation().	Responses if a process or environment is in simulation
cmdReplyList().	Parse message body to a list of reply parameters.
cmdFormatReply().	Formats reply buffer to an ascii string.
cmdReplyDefault().	Build a parameter structure filled with defaults from CDT.

Command Simulation




```
setenv CCSSTATE SIMULATION
OR ccsei menu OPTIONS Item SIMULATION
+
msgSend "" cmdManager SIMSET "<process name>"
OR ccsei menu WINDOWS, Item SIMULATION SETUP
```

Data Base(1)

- Data Types:
  - Integers (8,16,32 bits, (un)signed.
  - rfloat, double
  - Booleans
  - Bit fields,
  - Character strings fixed length
  - Time & Date
  - Internal Data Base References
- Structures:
  - Scalars,
  - Vectors (65535 elements of the same type)
  - Tables (Bi-Dimensional array, up to 65535 records (lines), up to 255 fields per record.

Data Base(2)

- Attributes grouped in points organized hierarchically, 0-255 attributes per point.
- Hierarchical view; Symbolic addressing: @<env>:point1:...:pointN.Attribute
- Distributed and Localized,
- Flat view: Symbolic: <alias>myPoint.Attribute but also Direct Address @(Plin.Ain)
- Current Working Point
- Implicit Open/Close
- Query, Read/Write, Multi-Read-Write
- dbBackup/Restore.
- Rtap Tools: DbConfig, PtSelector & Display
- ccsei tools: DbBrowser & DbMonitor

EVENTS:

- Notification of changes in the DB
- Message
- Attach
  - any write / filters (Scalars only)
  - optional user defined message
- Parsing:
  - old/new value for scalars
  - range for vectors/tables
  - Writing process

Logging System

The purpose of the system is :

- Log Information from local processes
- Collect and log errors generated locally or coming from a remote WS/LCU
- Monitoring and retrieval of logs.

The logging system supports two kinds of log :

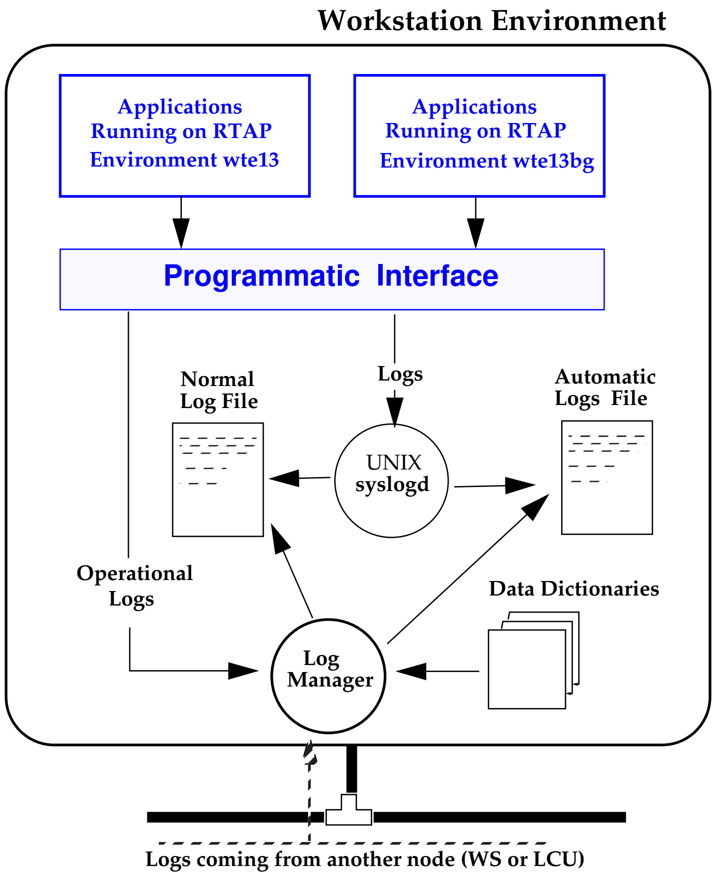
Normal Logs which are used to log information related to operative conditions. Among them, some **operational logs are logged in FITS format** and will represent the logbook of the telescope/instrument operation. These Fits logs are periodically retrieved, sorted by time, and archived.

Automatic Logs which are generated by specific modules and whose purpose is to help debugging and maintenance. For example the database and the message system can be configured to generate logs when specific activities take place. (Special commands MSGE/DLOG)

LogManager

The logManager is a process which connects to a list of LCU and waits for logs. Its main functions are :

- Configure the logging System at startup or on request.
- Collect logs from the LCUs
- Receive logs generated by the error system from other workstations.
- Support the logging of operational logs in FITS format and also access the data dictionaries to check log syntax.


Workstation Environment
Applications
Running on RTAP
Environment wte13
Applications
Running on RTAP
Environment wte13bg
Programmatic Interface
Normal
Log File
Logs
Automatic
Logs File
UNIX
syslogd
Operational
Logs
Data Dictionaries
Log
Manager
Logs coming from another node (WS or LCU)

Configuration

System Configuration

When the logging system is installed for the first time the **syslogd UNIX daemon** must be configured to log the data in the proper files.

Creation of the LCU reporting environment table

It is an ASCII file containing the association between LCU name and corresponding workstation reporting node.

The default LCU reporting environment table is a file common to all workstations and containing the configuration of all LCU/WS and it is stored in **$VLT_LOG_ROOT/logLCU.config.**

There can be configuration files dedicated for a specific environment : these files contain the name of the specific environment -i.e.- **logLCU.config.wte670**

Log clean-up procedure.

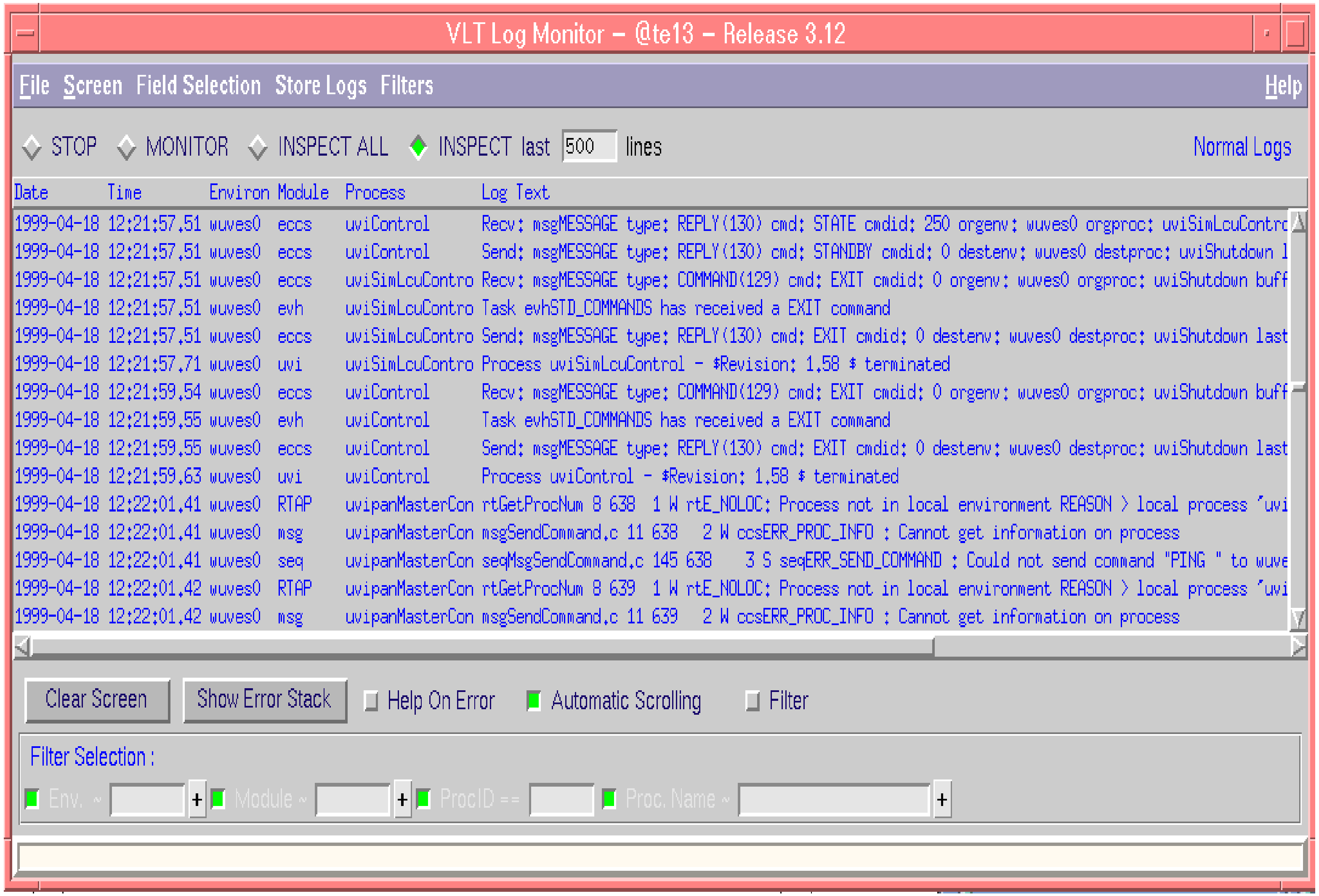
This is a script called **logVLTBackup** and running in background and activated by the cron UNIX utility.

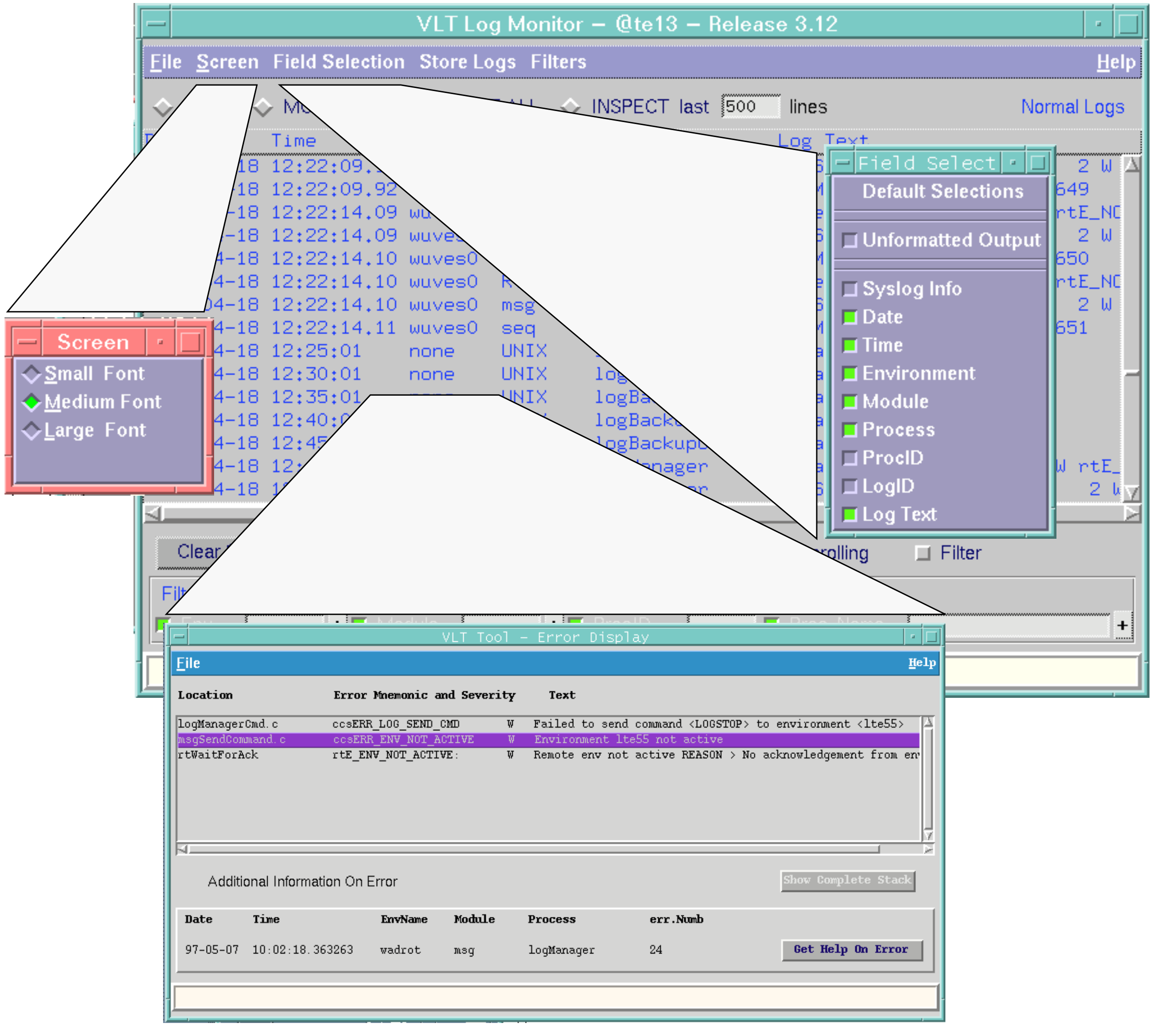
- Runs every 5 minutes.
- Checks the size of the VLT logFiles and if they exceed a given quantity it makes a backup copy and resets to zero-size the log files.

LogMonitor

The logMonitor is an Xwindows based utility written as a Tcl/Tk script which can be used (at WS level) to display and inspect the log messages.

- Monitor Mode : to display the incoming logs.
- Inspect Mode : Select and display the content of standard VLT log files.
- Selection of the format for displayed logs
- Provides filtering capabilities on what to be displayed.


VLT Log Monitor - @te13 - Release 3.12
File Screen Field Selection Store Logs Filters
Help
STOP MONITOR INSPECT ALL INSPECT last 500 lines
Normal Logs
Date Time Environ Module Process Log Text
1999-04-18 12:21:57.51 wuves0 eccs uviControl Recv: msgMESSAGE type: REPLY(130) cmd: STATE cmdid: 250 orgenv: wuves0 orgproc: uviSimLcuContro
1999-04-18 12:21:57.51 wuves0 eccs uviControl Send: msgMESSAGE type: REPLY(130) cmd: STANDBY cmdid: 0 destenv: wuves0 destproc: uviShutdown l
1999-04-18 12:21:57.51 wuves0 eccs uviSimLcuContro Recv: msgMESSAGE type: COMMAND(129) cmd: EXIT cmdid: 0 orgenv: wuves0 orgproc: uviShutdown buff
1999-04-18 12:21:57.51 wuves0 evh uviSimLcuContro Task evhSTD_COMMANDS has received a EXIT command
1999-04-18 12:21:57.51 wuves0 eccs uviSimLcuContro Send: msgMESSAGE type: REPLY(130) cmd: EXIT cmdid: 0 destenv: wuves0 destproc: uviShutdown last
1999-04-18 12:21:57.71 wuves0 uvi uviSimLcuContro Process uviSimLcuControl - $Revision: 1.58 $ terminated
1999-04-18 12:21:59.54 wuves0 eccs uviControl Recv: msgMESSAGE type: COMMAND(129) cmd: EXIT cmdid: 0 orgenv: wuves0 orgproc: uviShutdown buff
1999-04-18 12:21:59.55 wuves0 evh uviControl Task evhSTD_COMMANDS has received a EXIT command
1999-04-18 12:21:59.55 wuves0 eccs uviControl Send: msgMESSAGE type: REPLY(130) cmd: EXIT cmdid: 0 destenv: wuves0 destproc: uviShutdown last
1999-04-18 12:21:59.63 wuves0 uvi uviControl Process uviControl - $Revision: 1.58 $ terminated
1999-04-18 12:22:01.41 wuves0 RTAP uvipanMasterCon rtGetProcNum 8 638 1 W rtE_NOLOC: Process not in local environment REASON > local process 'uvi
1999-04-18 12:22:01.41 wuves0 msg uvipanMasterCon msgSendCommand.c 11 638 2 W ccsERR_PROC_INFO : Cannot get information on process
1999-04-18 12:22:01.41 wuves0 seq uvipanMasterCon seqMsgSendCommand.c 145 638 3 S seqERR_SEND_COMMAND : Could not send command "PING " to wuve
1999-04-18 12:22:01.42 wuves0 RTAP uvipanMasterCon rtGetProcNum 8 639 1 W rtE_NOLOC: Process not in local environment REASON > local process 'uvi
1999-04-18 12:22:01.42 wuves0 msg uvipanMasterCon msgSendCommand.c 11 639 2 W ccsERR_PROC_INFO : Cannot get information on process
Clear Screen
Show Error Stack
Help On Error
Automatic Scrolling
Filter
Filter Selection :
Env. ~
Module ~
ProcID ==
Proc. Name ~


VLT Log Monitor – @te13 – Release 3.12
File Screen Field Selection Store Logs Filters Help
INSPECT last 500 lines
Normal Logs
Screen
Small Font
Medium Font
Large Font
Field Select
Default Selections
Unformatted Output
Syslog Info
Date
Time
Environment
Module
Process
ProcID
LogID
Log Text
Filter
VLT Tool – Error Display
Location
Error Mnemonic and Severity
Text
Additional Information On Error
Show Complete Stack
Get Help On Error

Log API

- logData
- logFitsAction
- logFitsComment
- logFitsEvent
- loogIntParRecord
- logRealParRecord
- logStringParRecord
- logFitsParRecord
- logFitsParArray
- logFitsSetMask

Error System

The error system provides facilities to :

- Create and maintain the error definiton files
- log errors generated by local processes
- Provide tools to browse through the logged errors.

Set of related errors are grouped together under the same **error stack** and a **sequence number** defines the order of the error generation.

- **Error generated by processes running on a LCU are logged in the WS reporting node .**
- **Errors belonging to the same stack are logged on the same node (WS)**

Error Definition File

Each VLT software module must provide an ASCII file containing the definition of all errors possibly generated by the module.
Errors are defined in records with the following structure :

Error Number	The error number is always associated to a mnemonic which is used to retrieve the error message
Error Severity	Can have three values : **Warning, Serious and Fatal**.
Error Format	Describes the error message and the format of the parameters associated to the error, such as the printf format specification.
Help Reference	This is the name of the file containing the help text related to the specific error.

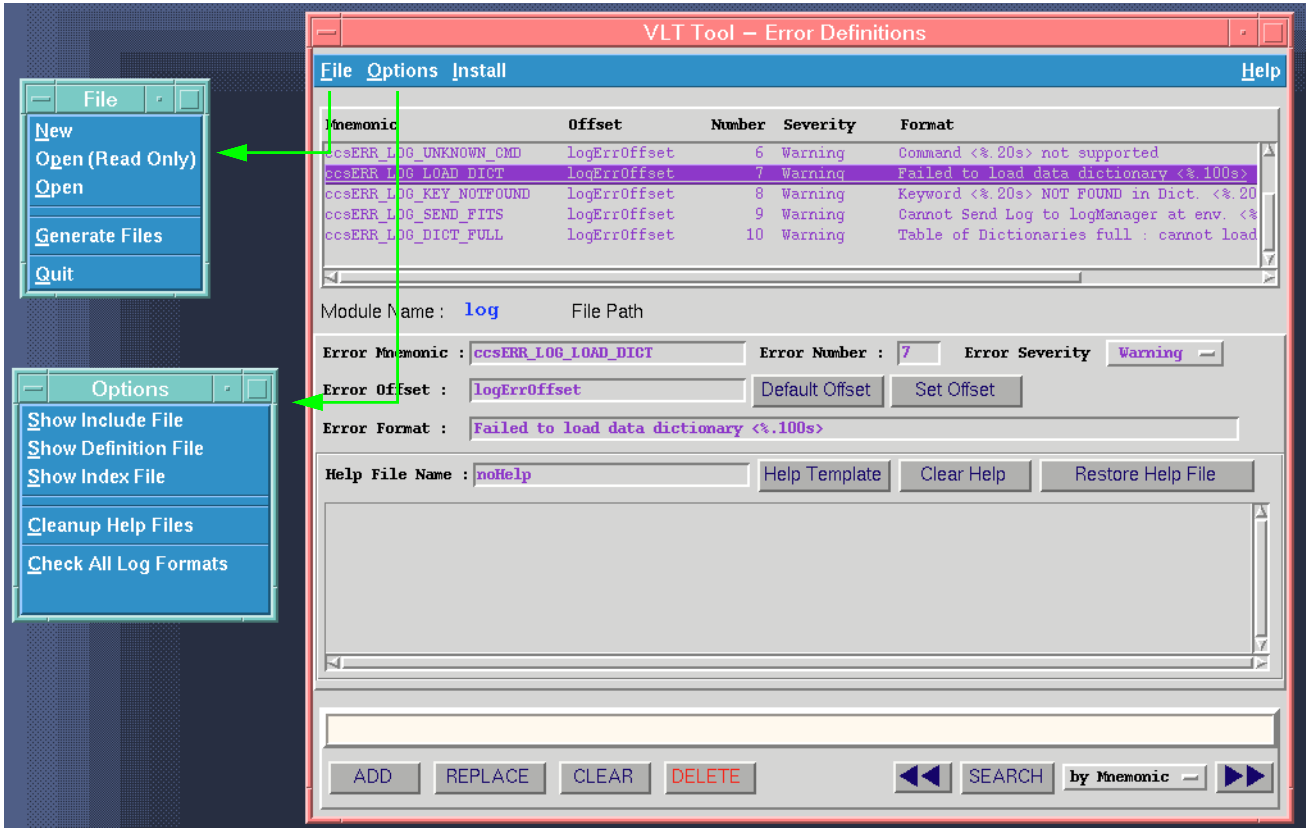
Example of error definition :

```
2 W
ccsERR_NOT_IMPL : Function %.40s not yet implemented
ccsERR_NOT_IMPL.hlp
```

Error Editor

The utility errEditor creates and modifies the error definition files : it generates three ASCII files :

<module>_ERRORS	File containing the error definitions (under ../ERRORS)
<module>Errors.h	Error Include File (under ../include)
<module>ERRORS.IDX	Error definition file (under ../ERRORS)



VLT Tool – Error Definitions
File Options Install
Help
File
New
Open (Read Only)
Open
Generate Files
Quit
Options
Show Include File
Show Definition File
Show Index File
Cleanup Help Files
Check All Log Formats
Mnemonic Offset Number Severity Format
ccsERR_LOG_UNKNOWN_CMD logErrOffset 6 Warning Command <%.20s> not supported
ccsERR_LOG_LOAD_DICT logErrOffset 7 Warning Failed to load data dictionary <%.100s>
ccsERR_LOG_KEY_NOTFOUND logErrOffset 8 Warning Keyword <%.20s> NOT FOUND in Dict. <%.20
ccsERR_LOG_SEND_FITS logErrOffset 9 Warning Cannot Send Log to logManager at env. <%
ccsERR_LOG_DICT_FULL logErrOffset 10 Warning Table of Dictionaries full : cannot load
Module Name : log
File Path
Error Mnemonic : ccsERR_LOG_LOAD_DICT
Error Number : 7
Error Severity Warning
Error Offset : logErrOffset
Default Offset
Set Offset
Error Format : Failed to load data dictionary <%.100s>
Help File Name : noHelp
Help Template
Clear Help
Restore Help File
ADD
REPLACE
CLEAR
DELETE
SEARCH
by Mnemonic

Programmatic Interface

The programmatic interface consist of the following routines.

errAdd	Add the current error condition to the error stack.
errSysAdd	Add to the stack the errors returned by RTAP and UNIX routines and the current error condition.
errResetStackK	Initialized the error structure and resets the stack **The information contained in the stack is not logged**
errConvRtapErr	Converts RTAP error numbers into CCS equivalent error.
errCloseStack	The current stack is logged and the error structure initialized.
errDisplay	Calls the error dialog to notify the user about an application error
errIsReason	Looks for a specific error in the error stack
errLccToCcs	Converts LCC errors to equivalent CCS errors
errPrint	Prints on the screen the content of the error condition
errMergeStack	Merge the stack associated to two different error structures into a single one

TIME System

- UTC/ST/JD calculations
- Time based actions:
  - Scheduling
  - Messages cmd/reply
- NTP (3.3) for CPU clock synchronisation.
- Standard HP-UX(10.10)
- ISO standard formats; old functions obsolete with OCT99

ALARM SYSTEM:

- Abnormal events/state machine
- Macros in dbl
  - analog
  - discrete
- Alarm display
  - state
  - location
  - user message
- Help on alarms
- Acknowledgement
- filtering
- FULL CCS ONLY.

SAMPLING:

- LCU data
- Rtap plotting
- more tolerant to CPU load
- integration WS sampling:
  – dbl classes for Rtap history tables.
  – Extended control panel.
- FULL CCS ONLY.

BOOKING SYSTEM:

- Based on Rtap Security mechanism.
- Controls user access to database & processes
- Configured by Access & Configuration Control
- Components: Processes or hardware units
  - They must be booked or defined as sharable to be controlled.
- Categories: (dbl)
  - VLT General
  - CCS administration
  - Instrument1
  - Telescope2
- User groups: privileges for commands.
  - Observer: public
  - Operator: public, maintenance
  - Developer: public, maintenance &test
  - BookMaster.
- FFULL CCS ONLY

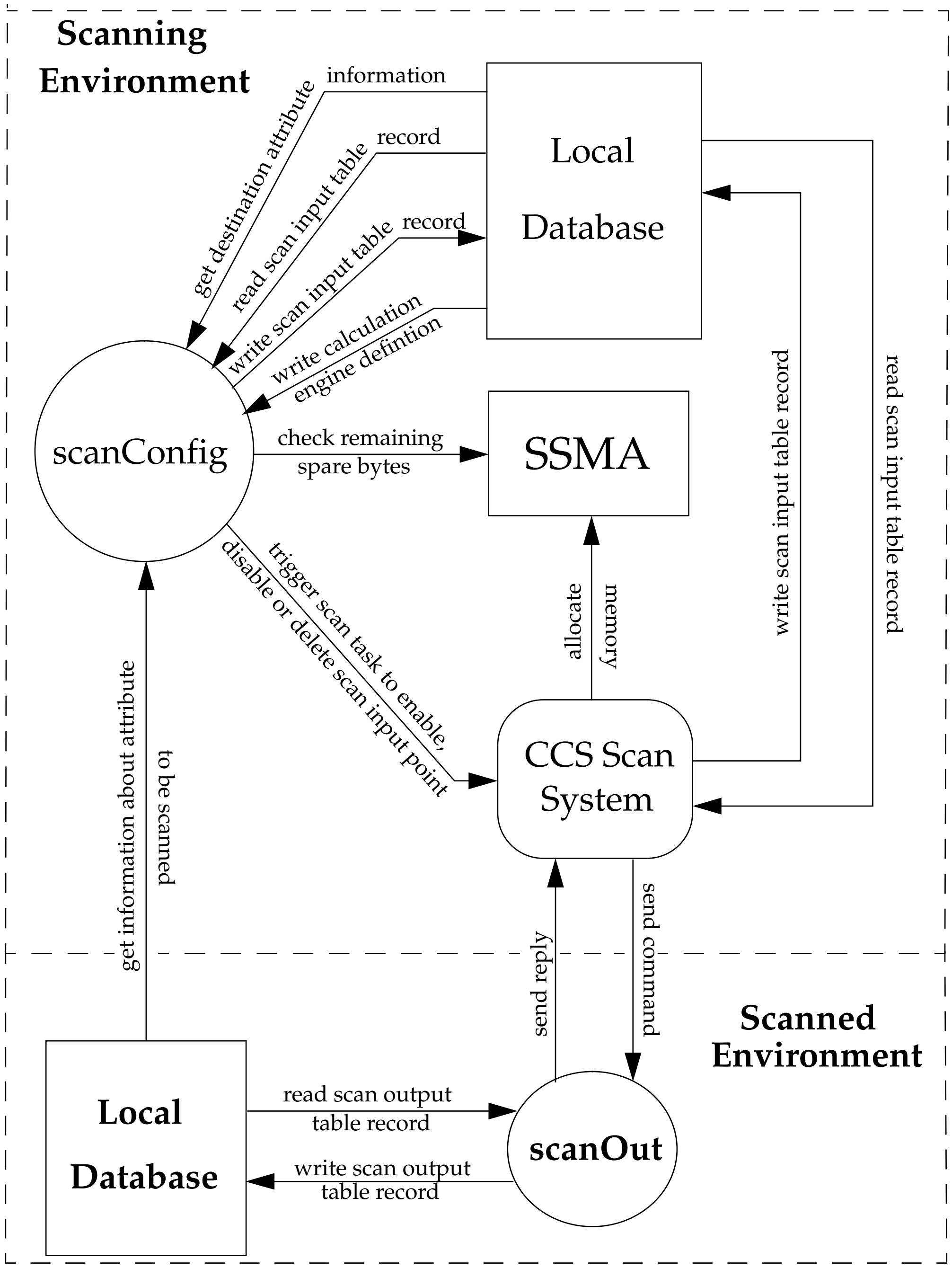
CCS

CCS: Scan System

SCANSYSTEM:

- DATA Propagation in DB:
  - 1 LCU->1 WS<->n WS
- POLL / SRBX
- Optimized writing
- scanLinks replacing scanConfig
- Auto restart.
- scanei
- CCS_Lite supports only scanning data towards a full CCS environment. Full support for OCT99.

Scan Config:


Scanning Environment
Local Database
information
get destination attribute
record
read scan input table
record
write scan input table
write calculation engine defintion
scanConfig
check remaining spare bytes
SSMA
allocate memory
write scan input table record
read scan input table record
trigger scan task to enable, disable or delete scan input point
CCS Scan System
get information about attribute to be scanned
send reply
send command
Scanned Environment
Local Database
read scan output table record
write scan output table record
scanOut

CCS Scan System DB Structure:

'root point' *comm-port level* *scan device level* *scan input point level*

- scan config (point)
  - LAN (point)
    - <scan device 1>(point)
      - scan input table (table attribute)
    - <scan device 2>
      - scan input table
    - <scan device n>
      - scan input table

The VLT SW distribution contains for each of these levels a class definition to be used for the database design (see file **/vlt/dbl/CCS.class**). These are

- "**<scan config>**" for the scan system root point
- "**<VLT comm-port>**" for a comm-port point
- "**<VLT scan dev>**" for a scan device point

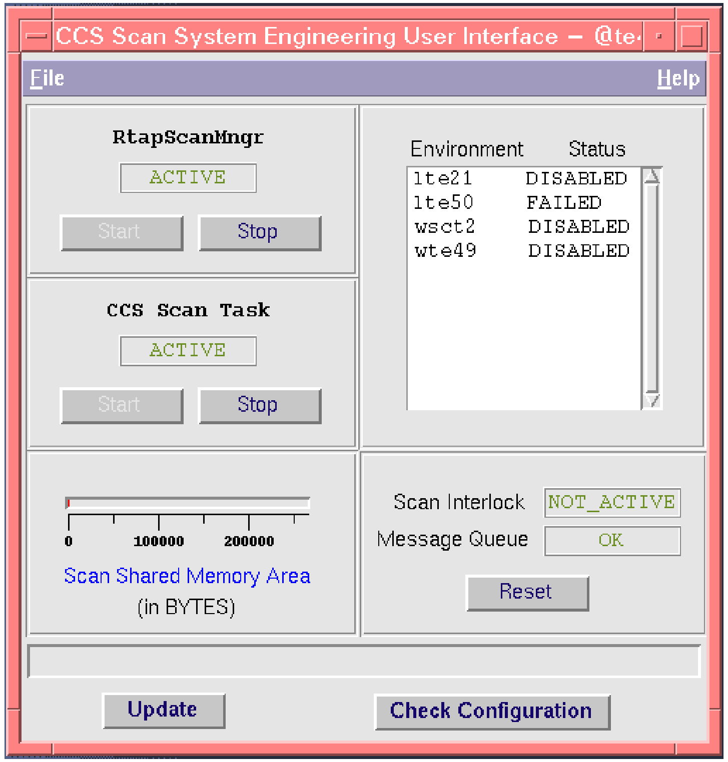
SCAN MODES:

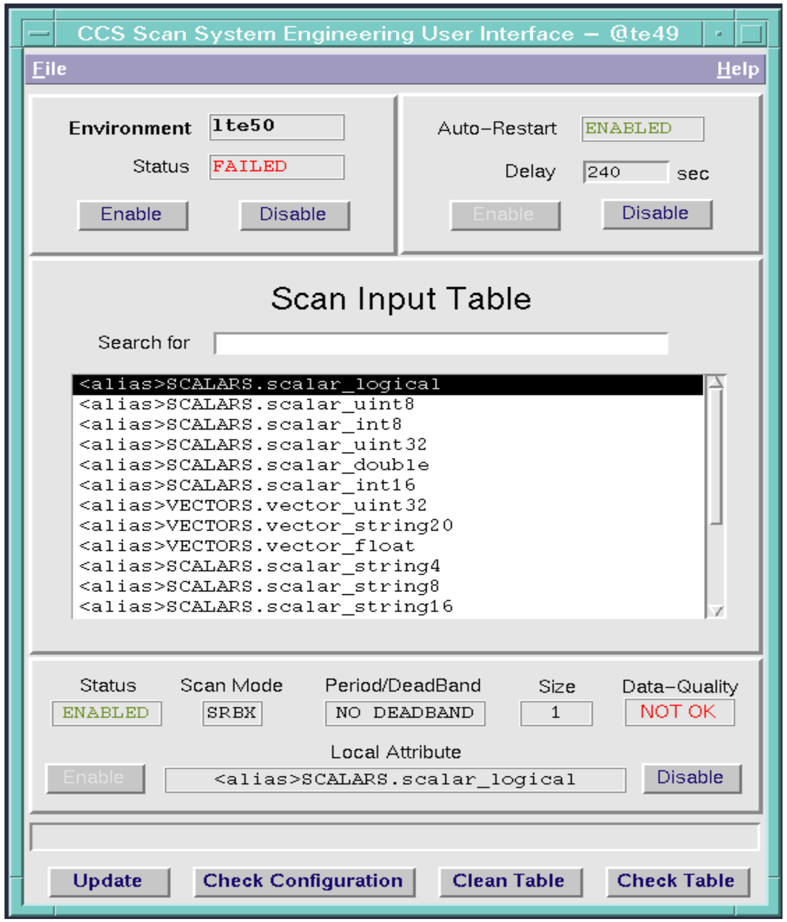
With the current version of the CCS Scan System attributes can be scanned in the following way:

	MODES	DEADBAND-TYPE	DESTINATION ATTRIBUTE
SCALARS	SRBX/POLL	0 or 1 (numerical) 2 (all types)	SCALAR
VECTORS			
1 record	SRBX/POLL 2 (all types)	0 or 1 (numerical)	SCALAR
several records	SRBX/POLL	2	VECTOR
TABLES			
1 field	SRBX/POLL	0 or 1 (numerical) 2 (all types)	SCALAR
1 column	SRBX/POLL	2	VECTOR

- each source attribute must be convertible into its destination attribute
-

The CCS Scan System Engineering User Interface

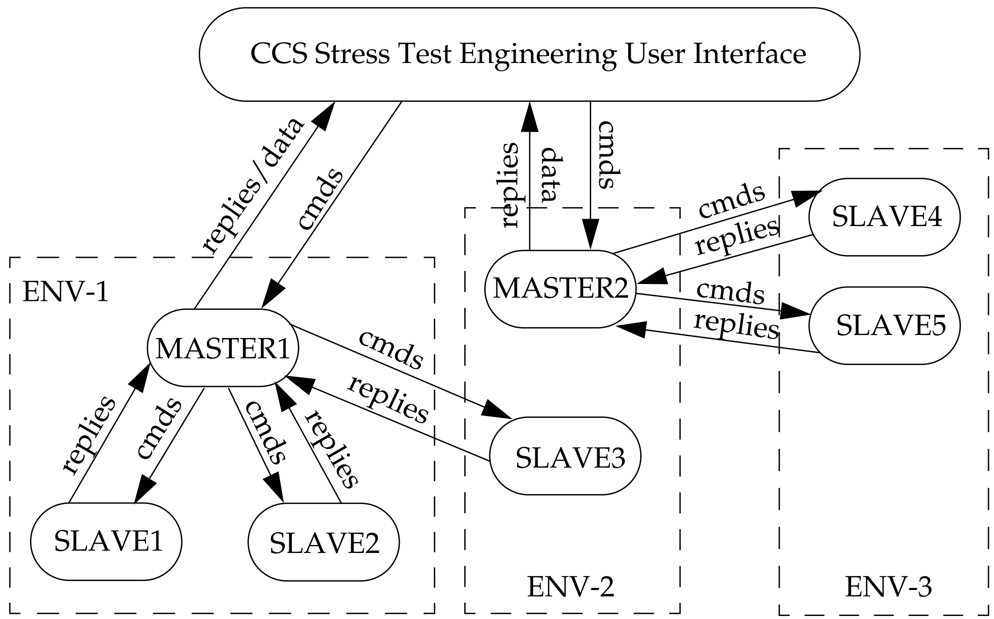




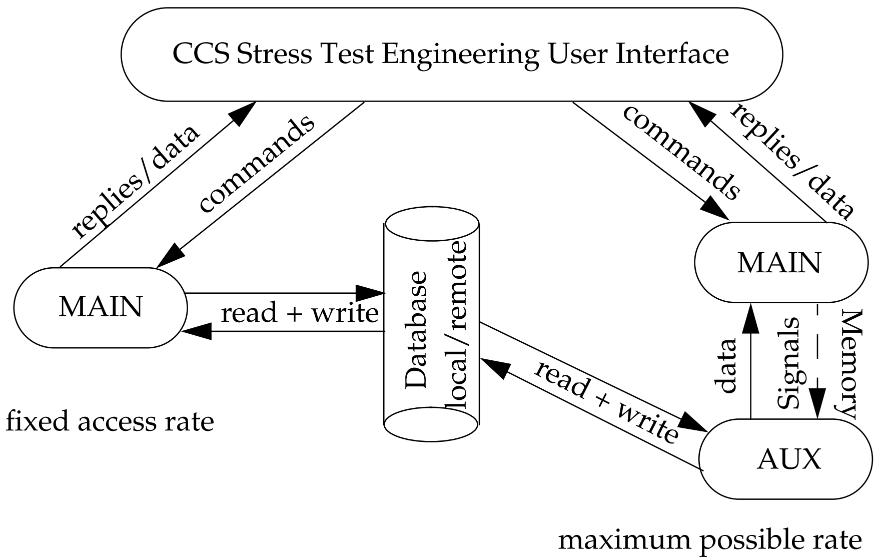
CCS

DATA BASE OPTIMIZATION:
CCS STRESS TESTS

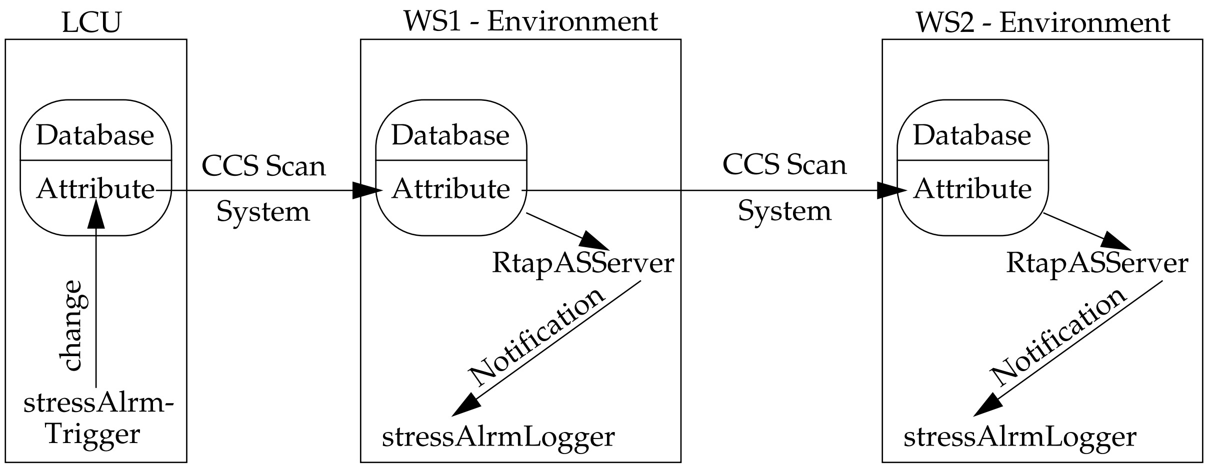
CCS Stress Tests: msg



CCS Stress Tests: Data Base



CCS Stress Tests: Alarms



CCS System Performances

Table 1: CCS Message System

	Message Rate (commands/sec)		Data Rate (KB/sec)	
	128 Bytes[a]	8000 Bytes	128 Bytes	8000 Bytes
Local - Local WS	894.5	640.3	111.8	5002.6
Local - Remote WS	232.0	88.5	29.0	691.4
Local WS - LCU	113.6	62.3	14.2	486.9
LCU -LCU	238.0	184.1	29.7	1438.8

a. Size of reply body

Table 2: CCS On-Line Database

Attributes referenced by[a]	Symbolic Name		Alias		Direct Address		Multi-[b]	
Accesses/second	Read	Write	Read	Write	Read	Write	Read	Write
Local WS	9162	1272	10018	1310	38052	1566	5 x 6000	5 x 1100
Local - Remote WS	1055	646	1257	657	1508	716	5 x 832	5 x 580
Local WS - LCU	77	77	85	81	-----	-----	5 x 49	5 x 49

Table 2: CCS On-Line Database

Attributes referenced by[a]	Symbolic Name		Alias		Direct Address		Multi-[b]	
Accesses/second	Read	Write	Read	Write	Read	Write	Read	Write
Local LCU	1142	1138	1468	1457	17964	16486	5 x 193	5 x 210

a. Test attributes are of typeSCALAR
b. List included 5 SCALARs

Table 3: CCS Alarm System

	LCU - WS1	WS1 - WS2	LCU - WS2
delay (sec) of notification	0.268	0.288	0.556

Table 4: CCS Scan System Part I

Number of Attributes/ Amount of Data[a]	CPU-LOAD ccsScan (%)	CPU-Load scanOut (%)[b]
100/039KB	0.44	0.63
300/1.56KB	0.83	1.74
600/3.12KB	1.39	3.46

a. The attributes are SCALARS scanned with a poll period of 1 second

b. The performance measured for **scanOut** running on a LCU is nearly the same

Table 5: CCS Scan System Part II

Number of attributes changing[a]	CPU-LOAD ccsScan (%)	CPU-Load scanOut (%)	CPU-Load RtapMQDBM (%)
0	1.39	3.46	**0.05**
10	1.39	3.46	**0.74**
30	1.45	3.46	**2.02**
50	1.42	3.46	**3.27**

a. Each attribute is changed every second

Recomandations:

- scan remote data rather than repeated remote access.
- use multi read/write to optimize DB access.
- avoid message deadlocks: a process should always quickly read messages on its input queue and not let them accumulate.

Not Supported in CCS-Lite:

- Scanning towards CCS_Lite env: OCT99.
- Calculation engine in Database,
- Data Quality
- Alarms
- Sampling, plotting, trending, history
- Booking, No DB Security.
- Events from CCS-Lite towards full CCS.
- Tims/Timer with remote envs.
- Rtap Tools: DbConfig, Report generator,...

and some other minor features as indicated in the CCS User Manual.