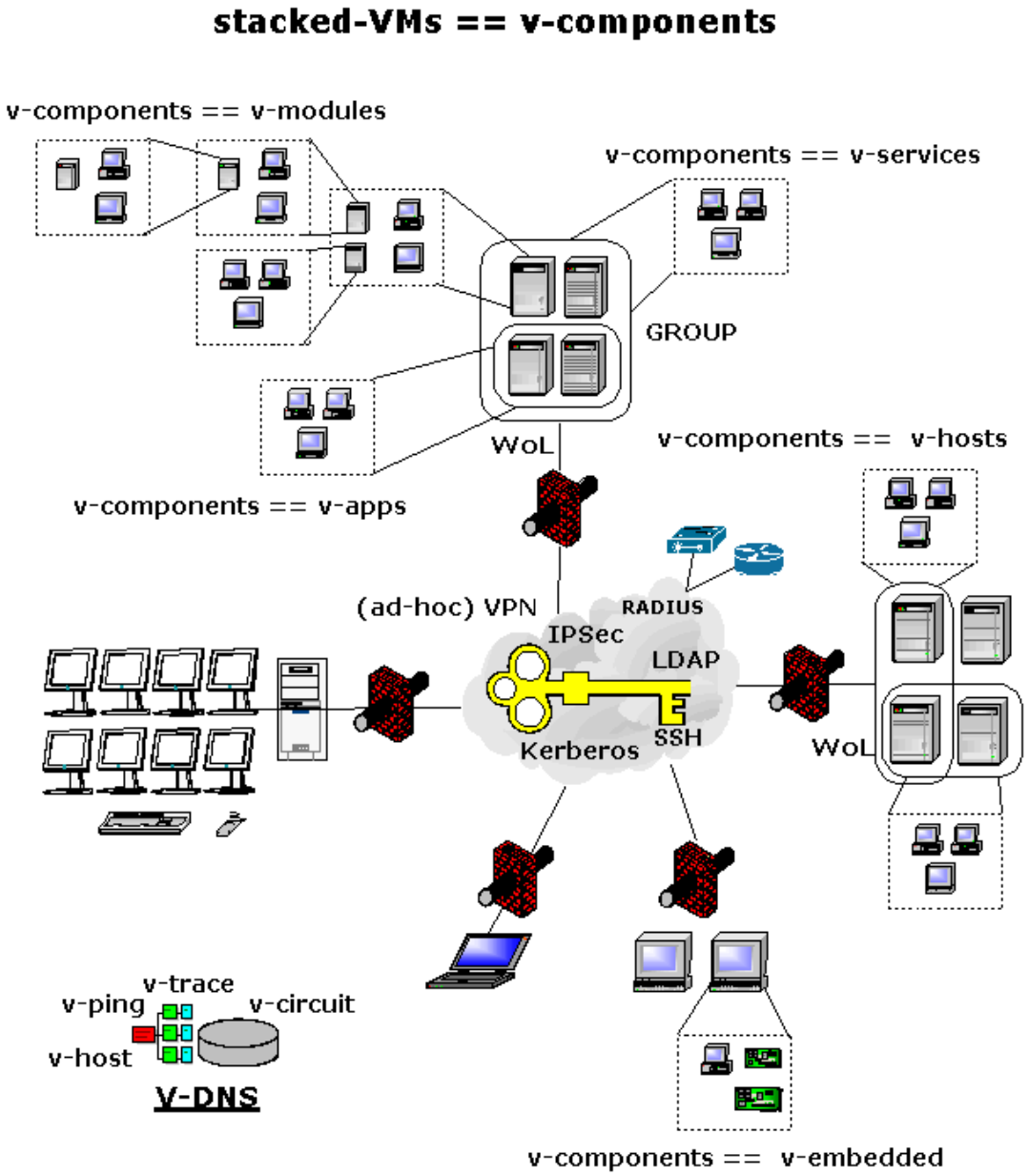

The UnifiedSessionsManager

User-Manual(draft-pre-release)
PDF for Online Help

Version:01.07.001b05a - 2009.01.26

Copyright 2008 by
Arno-Can Uestuensoez

Contents

I Common Basics **21**

1 Preface **23**
- 1.1 History . . . 23
- 1.2 Contact . . . 24
- 1.3 Some General Remarks . . . 24
- 1.4 Legal . . . 24
- 1.5 Acknowledgements . . . 25

2 Abstract **27**

3 Feature Specification **35**
- 3.1 Feature Introduction . . . 35
- 3.2 Feature-Sum-Up . . . 38

4 First-Time Implementations **45**
- 4.1 First set - 2008.02.11 . . . 45
- 4.2 Second set - 2008.07.10 . . . 48

5 Secure Sessions **51**
- 5.1 Security and Authorization . . . 53
- 5.2 Xinerama Screen Layouts . . . 54
  - 5.2.1 Physical Layout-1 . . . 55
  - 5.2.2 Physical Layout-2 . . . 60
- 5.3 Client-Session Windows . . . 61
- 5.4 Bulk Access . . . 64
- 5.5 Encryption and Tunneling with SSH . . . 67
  - 5.5.1 DISPLAYFORWARDING . . . 69
  - 5.5.2 CONNECTIONFORWARDING . . . 70
  - 5.5.3 Execution-Locations . . . 72

6 Advanced Features **73**
- 6.1 Bulk Access . . . 73
- 6.2 CLI-MACROS . . . 73
- 6.3 Generic Custom Tables . . . 74
- 6.4 Parallel and Background Operations . . . 74

7 HOSTs - The Native Access to OS **77**
7.1 Command Line Access - CLI . . . 77
7.2 Start a GUI application - X11 . . . 78
7.3 Open a complete remote Desktop - VNC . . . 78

8 PMs and VMs - The Stacked-Sessions **81**
8.1 Session-Types . . . 84
8.2 VM-Stacks - Nested VMs . . . 87
8.2.1 Stacked-Operations . . . 87
8.2.2 Specification of VM Stacks . . . 88
8.2.3 Bulk-Core CPUs . . . 94
8.2.4 Almost Seamless Addressing . . . 94
8.3 Stacked Networking . . . 95
8.4 Stacked Functional Interworking . . . 97
8.4.1 Stack-Address Evaluation . . . 97
8.4.2 Startup . . . 98
8.4.3 Shutdown . . . 99
8.4.4 State-Propagation Basics . . . 102

9 CTYS-Nameservices **109**
9.1 Basics . . . 109
9.2 Runtime Components . . . 115
9.2.1 Distributed Nameservice - CacheDB . . . 118
9.2.2 Network LDAP-Access . . . 121
9.2.3 Application Range and Limits . . . 121
9.3 Required Namebinding . . . 121
9.3.1 Integration of PMs, VMs, and HOSTs . . . 122
9.3.2 Basics of Name Bindings . . . 123
9.4 Group-Targets . . . 124

II Software Design **127**

10 Software Architecture **129**
10.1 Basic Modular Design . . . 129
10.2 Development Environment . . . 131
10.2.1 Some basics on "bash" . . . 131
10.2.2 Component Framework . . . 132
10.3 All about bash-Plugins and bash-Libraries . . . 136
10.3.1 The Differences . . . 136
10.3.2 Libraries . . . 136
10.3.3 Plugins . . . 136
10.4 ctys Control and Data Flow . . . 139
10.4.1 Distributed Controller . . . 139
10.4.2 Task Data . . . 140
10.4.3 Stack Interworking . . . 140

11 Interface Design **145**
11.1 Target-Platforms . . . 145
11.2 Feature-Design . . . 145
11.2.1 Communications Modes . . . 145
11.3 Command-Sets . . . 148
11.4 Plugins . . . 148
11.4.1 Main Dispatcher . . . 148
11.4.2 Standard Plugins . . . 148
11.4.3 Extensions . . . 148
11.4.4 Subdispatcher . . . 148
11.5 Security Design . . . 153
11.5.1 Accounts Impersonation . . . 153
11.5.2 Access Location . . . 153
11.5.3 System Resources . . . 153

12 CTYS-Nameservices **155**
12.1 Runtime Components . . . 155

III User Interface **157**

13 Common Syntax and Semantics **159**
13.1 General CLI processing . . . 159
13.2 Options Scanners - Reserved Characters . . . 161
13.3 Hosts, Groups, VMStacks and Sub-Tasks . . . 161
13.3.1 Common Concepts . . . 161
13.3.2 Flat Execution-Groups by Include . . . 164
13.3.3 Structured Execution-Groups by Sub-Tasks . . . 167
13.3.4 Stacks as Vertical-Subgroups . . . 169
13.3.5 VCircuits as Sequentially-Chained-Subgroups . . . 176
13.4 CLI macros . . . 177
13.5 Shell Interworking . . . 180
13.5.1 ctys as Sub-Call . . . 180
13.5.2 ctys as Master-Call . . . 180
13.5.3 Generic Remote Execution . . . 180
13.5.4 Access to ctys Functions . . . 180
13.6 Common Options . . . 180

14 Core Data **183**
14.1 Overview . . . 183
14.2 Standard Configuration Files . . . 184
14.3 Common Data Fields . . . 186
14.4 Common Processing Options . . . 196
14.5 Specific Variations . . . 201
14.6 Generic Tables . . . 202

15 Address Syntax **205**
15.1 Basic Elements . . . 205
15.2 Syntax Elements . . . 206
15.3 Stack Addresses . . . 212
15.4 Groups of Stack Addresses . . . 213

16 CTYS Call interface **215**
16.1 Common Elements and Behaviour . . . 215
16.1.1 Sessions Namebinding . . . 215
16.1.2 Cache for Permormance and Collections . . . 215
16.1.3 Access VM Stacks . . . 217
16.2 ctys Actions . . . 220
16.3 ctys Generic Options . . . 247
16.4 ctys Arguments . . . 270
16.5 ctys-plugins . . . 272

17 Plugins - Feature Extensions **273**
17.1 Prerequisites . . . 273
17.2 Category HOSTs . . . 276
17.2.1 CLI - Command Line Interface . . . 276
17.2.2 X11 Interface . . . 280
17.2.3 VNC - Virtual Network Console Interface . . . 283
17.3 Category PMs . . . 289
17.3.1 PM - Linux and BSD Hosts . . . 289
17.4 Category VMs . . . 300
17.4.1 QEMU - QEMU Emulator . . . 300
17.4.2 VMW - VMware . . . 314
17.4.3 XEN - Xen . . . 319
17.5 Category GENERIC . . . 326
17.5.1 LIST Plugin-Collector . . . 326
17.5.2 ENUMERATE Plugin-Collector . . . 326
17.5.3 SHOW Plugin-Collector . . . 326
17.5.4 INFO Plugin-Collector . . . 326
17.6 Category Internal . . . 327
17.6.1 DIGGER - Forwarding Encrypted Connections . . . 327

18 Support Tools **329**
18.1 ctys-dnsutil . . . 330
18.2 ctys-extractAPRlst . . . 334
18.3 ctys-extractMAClst . . . 337
18.4 ctys-genmconf . . . 340
18.5 ctys-groups . . . 344
18.6 ctys-macros . . . 346
18.7 ctys-macmap . . . 348
18.8 ctys-plugins . . . 352
18.9 ctys-setupVDE . . . 356
18.10ctys-smbutil . . . 364
18.11ctys-vdbgen . . . 366
18.12ctys-vhost . . . 372
18.13ctys-vping . . . 397

18.14ctys-wakeup . . . 400
18.15ctys-xen-network-bridge . . . 403
18.16ctys-utilities . . . 406
18.16.1 ctys-install . . . 406
18.16.2 ctys-install1 . . . 409
18.16.3 getCurOS . . . 410
18.16.4 getCurOSRelease . . . 410
18.16.5 getCurDistribution . . . 410
18.16.6 getCurRelease . . . 410
18.16.7 pathlist . . . 410
18.16.8 ctys-getMasterPid . . . 410

IV Examples 411

19 General Remarks 413

20 ctys Setup 415
20.1 Installation . . . 415
20.1.1 Basic Install . . . 415
20.1.2 Security Environment . . . 421
20.1.3 Setup Access Permissions . . . 423
20.2 Configuration . . . 424
20.2.1 Plugins . . . 424

21 HOSTs - Sessions 429
21.1 CLI Examples . . . 429
21.1.1 Single Local Interactive Session . . . 429
21.1.2 Single Remote Interactive Session . . . 430
21.1.3 Execute a Remote Command . . . 430
21.1.4 Execute Multiple Remote Commands . . . 431
21.1.5 Chained Logins by Relays . . . 431
21.1.6 Xterm with tcsh . . . 432
21.1.7 gnome-terminal . . . 432
21.1.8 Call ctys functions . . . 433
21.2 X11 Examples . . . 437
21.2.1 Xterm with Interactive bash . . . 437
21.2.2 Gnome-Terminal with Interactive bash . . . 438
21.2.3 Emacs Session in shell-mode . . . 439
21.2.4 Single XClock . . . 439
21.3 VNC Examples . . . 439
21.3.1 Single Local Desktop Session . . . 440
21.3.2 Single Remote Desktop Session . . . 440
21.3.3 Bulk Desktop Sessions . . . 441
21.3.4 Remote Desktop with Local Client . . . 442
21.3.5 Shared-Mode vs. Non-Shared-Mode . . . 444
21.3.6 Reconnect Suboption . . . 444

22 VMs - Sessions **445**
22.1 QEMU Examples . . . 445
22.1.1 Installation of QEMU . . . 445
22.1.2 Installation of Guests . . . 452
22.1.3 CREATE a session . . . 456
22.1.4 CANCEL a session . . . 459
22.1.5 LIST sessions . . . 459
22.1.6 ENUMERATE sessions . . . 460
22.1.7 SHOW . . . 460
22.1.8 INFO . . . 460
22.1.9 Example Appliances . . . 463
22.2 VMW Examples . . . 464
22.2.1 Installation of VMW . . . 464
22.2.2 Installation of Guests . . . 465
22.2.3 CREATE a session . . . 469
22.2.4 CANCEL a session . . . 471
22.2.5 LIST sessions . . . 472
22.2.6 ENUMERATE sessions . . . 474
22.2.7 SHOW . . . 475
22.2.8 INFO . . . 475
22.2.9 Display of Available Sessions . . . 475
22.2.10 Example Appliances . . . 479
22.3 Xen Examples . . . 479
22.3.1 Installation of Xen/Dom0 . . . 479
22.3.2 Installation of Guests/DomU . . . 479
22.3.3 CREATE a session . . . 483
22.3.4 CANCEL a session . . . 492
22.3.5 LIST sessions . . . 496
22.3.6 ENUMERATE sessions . . . 498
22.3.7 SHOW . . . 499
22.3.8 INFO . . . 500
22.3.9 Example Appliances . . . 500

23 PMs - Sessions **501**
23.1 PM Examples . . . 501
23.1.1 Configuration of Access Permissions . . . 501
23.1.2 CREATE a PM Session . . . 502
23.1.3 CANCEL a PM Session . . . 503
23.1.4 PM - Using Wake-On-Lan - WoL . . . 505
23.1.5 Wake-On-Lan - Complete Reboot-Cycles . . . 511

24 Common Session Options **517**
24.1 Opening multiple ctyss . . . 517
24.1.1 Multiple Calls . . . 517
24.1.2 One call . . . 517
24.2 Different resolution on client and server . . . 518
24.3 Dynamic move of sessions window . . . 518
24.4 Spanning Multiple Physical Screens . . . 522
24.5 CREATE with tree-search for unique IDs . . . 524
24.6 Some session related calls . . . 525

24.6.1 ENUMERATE . . . 525
24.6.2 LIST . . . 525
24.6.3 SHOW . . . 525
24.6.4 INFO . . . 526
24.7 Some ctys related calls . . . 526
24.7.1 Display Version and available plugin . . . 526

25 Custom CLI **527**
25.1 Groups . . . 527
25.2 Macros . . . 529
25.3 Tables . . . 529
25.3.1 Common Tables for ENUMERATE and LIST . . . 529
25.3.2 RAW Tables by MACHINE . . . 531
25.3.3 Combined MACROS and Tables . . . 531

26 Pre-Configured Desktops **543**
26.1 Admin RAID-Info . . . 543
26.2 Tax-Calculation and Longterm-Storage . . . 547
26.3 QEMU Test-Environment for VDE . . . 551

27 Performance Measures **555**
27.1 Background, Parallel and Cached Operations . . . 555
27.2 Caching of Data . . . 556
27.3 Combined Operations . . . 557

28 VM Stacks **559**
28.1 Setup of the Stack Environment . . . 559
28.2 Single-Upper-Layer VMStack Basics . . . 563
28.2.1 CREATE a PM Session . . . 563
28.2.2 CREATE a VM Session with initial PM Session . . . 564
28.2.3 CREATE a HOST Session with initial VM and PM Session 564
28.2.4 CANCEL a VM Session containing VM-Stack-Sessions . . . 564
28.2.5 CANCEL a PM Session containing VM-Stack-Sessions . . . 564
28.3 Multi-Layer VMStacks . . . 564
28.3.1 MACRO Definition Basics . . . 564
28.3.2 CREATE a VMStack . . . 564
28.3.3 CANCEL a VMStack . . . 564
28.3.4 Prepare a PM for Restart by WoL . . . 564
28.3.5 Xen with upper QEMU-VMs . . . 565
28.3.6 VMWare with upper QEMU-VMs . . . 571
28.3.7 QEMU with upper QEMU-VMs . . . 571

29 User Access - Secure permissions **573**
29.1 SSH, SSL, IPsec, and VPNs . . . 573
29.2 VM-Stacks, SSO, and Kerberos . . . 574
29.3 Network Filesystems . . . 574
29.4 Authentication - ksu and sudo . . . 574
29.4.1 Basics . . . 574
29.4.2 sudoers . . . 574
29.4.3 k5users . . . 576

29.4.4 Secure root Access . . . 577
29.5 Firewall . . . 577
29.6 SELinux . . . 577

30 System Resources **579**
30.1 TAP/TUN by VDE . . . 579
30.1.1 VDE within Xen Dom0 . . . 580
30.1.2 VDE within Xen DomU . . . 583
30.1.3 VDE within VMware . . . 583
30.1.4 VDE within Native Unix . . . 583
30.1.5 VDE Remote Configuration . . . 586
30.1.6 VDE-Setup with Micro-VMSTACK . . . 588
30.2 Mount an ISO-Image . . . 589

31 Applications of ctys-vhost **591**
31.1 Database Generation ctys-vdbgen and companions . . . 591
31.1.1 Initial Database . . . 592
31.1.2 Append Data to present Database . . . 596
31.1.3 vm.conf Coallocated with Hypervisor . . . 599
31.2 LDAP based VM-Stack Nameservices . . . 600
31.3 Group Addressing . . . 600
31.3.1 Group Caches - The Performance Clue . . . 600
31.3.2 Preconfigured Task-Groups . . . 600
31.4 Stack Addressing . . . 600
31.5 DHCP Poll . . . 600
31.6 PXELinux - Address Translation . . . 602
31.7 Formatted output by Generic Tables . . . 604
31.8 Filter by awk-Regular Expressions . . . 605
31.8.1 Datastreams and Match Conditions . . . 605
31.8.2 Wildcards and Ambiguity . . . 606

V Appendices **609**

32 Current Loaded Plugins **611**

33 File Formats **617**
33.1 Common Tools and Formats . . . 617
33.2 Groups . . . 618
33.3 Macros . . . 621
33.4 Static Import File Databases - fdb . . . 622
33.4.1 macmap.fdb . . . 622
33.4.2 enum.fdb . . . 622
33.5 Static Pre-Fetch Cache Databases - cdb . . . 623
33.5.1 grpscache.cdb . . . 623
33.5.2 statcache.cdb . . . 624
33.6 Dynamic Runtime Cache Databases . . . 624
33.7 Configuration Entries for ctys . . . 624
33.7.1 Actual Processing vs. Administrative Display . . . 624
33.7.2 Configuration File Variants . . . 625

33.7.3 Keywords . . . 627
33.7.4 Interface Keywords . . . 631
33.7.5 MAGICID . . . 634
33.7.6 VMSTATE . . . 634
33.7.7 Virtual Hardware-Platform . . . 634
33.7.8 Stacking Entries . . . 637
33.7.9 Execution Location and Relocation . . . 637

34 LDAP Formats **639**

35 Call Input-Output Formats **641**
35.1 Common Data Import/Export Options . . . 641
35.2 ENUMERATE . . . 641
35.3 LIST . . . 641
35.4 INFO . . . 641
35.5 SHOW . . . 641

36 Miscellaneous **643**
36.1 Basic EXEC principle . . . 643
36.2 PATH . . . 644
36.3 Configuration files . . . 645
36.4 Media Access Control(MAC) Addresses - VM-NICs . . . 645

37 GNU GENERAL PUBLIC LICENSE - Version 3 **651**

Bibliography **669**
Books . . . 669
UNIX . . . 669
Security . . . 670
Networks . . . 672
Embedded Systems . . . 673
Online References . . . 674
OSs . . . 674
Hypervisors/Emulators . . . 675
Security . . . 677
Specials . . . 677
Miscellanuous . . . 678
UnfiedSessionsManager Versions . . . 678
Sponsored OpenSource Projects . . . 680
Commercial Support . . . 680

List of Tables

3.1 Tested OS support for Stacked-VMs on PC-Platform 40
3.2 Tested Hypervisor Versions 41
3.3 Tested Native Plugins vs. OS-Distribution 42
3.4 Planned roadmap for new features(N:next vers.) 43

5.1 Mapping schema of labels to screens 59

8.1 List of Standard Plugins 85
8.2 Targets for state propagation of CANCEL action 102
8.3 State-Propagation for the first version 104
8.4 Application of Propagation Scopes 105

11.1 ENUMERATE-Input-Format from Plugins 150
11.2 ENUMERATE-Output-Format of Sub-Dispatcher 151
11.3 LIST-Input-Format from Plugins 152
11.4 LIST-Output-Format of Sub-Dispatcher 152

14.1 Supported File-Extensions 185

16.1 Supported Boot-Modes for XEN 227
16.2 Supported Boot-Modes 228
16.3 Supported Console-Types 230
16.4 Supported Command-Execution 231
16.5 Supported Debugger 231
16.6 Processing suboptions 238
16.7 Output Record-Format for MACHINE suboption 239
16.8 Output-Format for MACHINE suboption 245
16.9 Processing suboptions 245

17.1 Prerequisites and applicability of plugins 273
17.2 Utilized QEMU-Monitor-Commands 307
17.3 Supported Configuration files for QEMU 310
17.4 Forwarding modes and call locations for VMW versions 318
17.5 Applicable forwarding modes and call locations for XEN 322

22.1 Overview of Installed-QEMU-VMs 455
22.2 Overview of Intsalled-VMs 469
22.3 Overview of Intsalled-Xen-VMs 483

26.1 QEMU Base Tests for "ctys-setupVDE" 551

27.1 Performance effects of "-b" option 557
27.2 Performance effects of "-b" option 558

List of Figures

2.1 The UnifiedSessionsManager . . . 27
2.2 Physical Multi-Monitor Design . . . 29
2.3 Logical Xinerama-Mode . . . 30
2.4 XEN Hypervisor/Emulator and CentOS Host within EMACS . . . 32
2.5 QEMU Hypervisor/Emulator and CentOS Host within EMACS . . . 32

5.1 Physical Multi-Monitor Layout-1 . . . 55
5.2 Physical Multi-Monitor Design . . . 56
5.3 Logical Multi-Screen Layout-1 . . . 56
5.4 Logical Xinerama Layout . . . 57
5.5 Logical Multi-Screen X11-Remapping . . . 58
5.6 Logical Multi-Screen X11-Array-Style . . . 58
5.7 Physical Multi-Monitor Array-Style Addressing . . . 59
5.8 Mapping schema for multiple Desktops/Workspaces . . . 60
5.9 Physical Multi-Monitor Layout-2 . . . 61
5.10 Basic handling of client sessions windows . . . 62
5.11 DISPLAYFORWARDING . . . 69
5.12 CONNECTIONFORWARDING . . . 70

8.1 Supported Stackmodels . . . 83
8.2 Pane-View: QEMU-ARM in Xen-DomU . . . 89
8.3 Stack-View: QEMU-ARM in Xen-DomU . . . 90
8.4 Pane-View: W2K in VMware on Linux . . . 91
8.5 Stack-View: W2K in VMware on Linux . . . 91
8.6 Pane-View: Virtual PC with Linux in VMware . . . 92
8.7 Stack-View: Virtual PC with Linux in VMware . . . 93
8.8 Virtual interconnection structure . . . 96
8.9 Nested Protocol Stacks . . . 97
8.10 Stack-View: W2K as HVM in DomU . . . 100

9.1 Stack-Controller Data . . . 110
9.2 Stack-Controller Data Visibility . . . 113
9.3 Nameservice components . . . 116
9.4 Cache Generation . . . 119
9.5 Distributed Caches . . . 120

10.1 ctys Software Layers . . . 129
10.2 ctys Local Control Flow . . . 139

10.3 Task Data handled by the main dispatcher . . . 140
10.4 Nested Upward-Stackpropagation . . . 141

12.1 Nameservice components . . . 156

13.1 Subtask . . . 163
13.2 Groupresolution by Include only . . . 164
13.3 Groupresolution by Subgroups . . . 167
13.4 Combined Subgroups and Substacks . . . 170
13.5 Stack Example for Basic Call-Interface . . . 174
13.6 CONSOLE- and HOSTs-Asynchronity for Stacked-Execution . . . 175
13.7 VCIRCUIT . . . 177

15.1 TAE - Target Application Entity address . . . 207
15.2 Machine-Address . . . 207
15.3 TDE - Target Display Entity address . . . 207
15.4 TAE - Target Application Entity address . . . 208
15.5 Stack-Address . . . 212
15.6 Groups of Stack-Addresses . . . 213
15.7 Groups member option expansion . . . 214

17.1 Structure of WoL configuration . . . 290
17.2 QEMU NIC interconnection . . . 301
17.3 UnifiedSessionsManager QEMU file structure . . . 304
17.4 QEMU Interconnection-Interfaces . . . 306

20.1 Install Steps . . . 420

22.1 QEMU Interconnection-Interfaces . . . 451
22.2 The "CONSOLE:EMACSAM" for a QEMU Session . . . 458
22.3 The "CONSOLE:EMACSAM" and a "CONSOLE:VNC" for a QEMU Session . . . 459
22.8 VMware WS6 with an additional VNCviewer Client Session. . . . 471
22.18The "CONSOLE:EMACSAM" for a XEN Session . . . 488
22.24TAB_CPORT by LIST . . . 498
22.25TAB_CPORT by ENUMERATE . . . 499

23.1 "Bridge-less" WoL configuration within same Ethernet Segment . 507
23.2 WoL configuration with a virtual bridge . . . 508
23.3 WoL configuration with a virtual bridge on a bond device . . . 509
23.4 WoL configuration for a NIC behind a router . . . 510

26.1 Administration of RAID and Power-Supply . . . 544
26.2 Logical Multi-Screen X11-Array-Style . . . 544
26.3 Registration and Longterm Storage of receipts for TAX requirements . . . 548
26.4 PMs for basic tests of "ctys-setupVDE" . . . 552
26.5 Various PMs for expansion of test scope of "ctys-setupVDE" . . . 553

28.1 Example for a simple Stack . . . 559
28.2 Example for a Small Stacked-Network . . . 560

28.3 Filesystem Structure 562
28.4 Stacked VMs and Filesystem Access 562
28.5 Example for a Single Stack-Column 566
28.6 Example for a Single Stack-Column of Network Components . . 566
28.7 VMSTACK tst-01a in action. 570
28.8 LIST of running VM instances 571

30.1 Virtual switch by vde_switch 580
30.2 QEMU NIC interconnection 581
30.3 QEMU NIC interconnection for Native UNIX 583

31.1 TAB_CPORT by ctys-vhost 593
31.2 TAB_CPORT by ctys-vhost 594
31.3 TAB_CPORT by ctys-vhost 594
31.4 TAB_CPORT by ctys-vhost 595
31.6 TAB_CPORT by ctys-vhost 604
31.7 Default table for ctys-vhost: TAB_CTYS_VHOST_DEFAULT 605
31.8 ctys-vhost awk-regexpr-1 607
31.9 ctys-vhost awk-regexpr-1 607
31.10ctys-vhost awk-regexpr-1 608

Licence - GPL3

Copyright (C) 2007,2008 Arno-Can Uestuensoez (UnifiedSessionsManager.org)

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program. If not, see <http://www.gnu.org/licenses/>.

```
------------------------------------------------------------------------
PROJECT          = Unified Sessions Manager
------------------------------------------------------------------------
CALLFULLNAME     = Commutate To Your Session
CALLSHORTCUT     = ctys

AUTHOR           = Arno-Can Uestuensoez - acue@UnifiedSessionsManager.org
MAINTAINER       = Arno-Can Uestuensoez - acue_sf1@sourceforge.net
VERSION          = 01_07_001b02
DATE             = 2008.08.06

COPYRIGHT        = Arno-Can Uestuensoez - acue@UnifiedSessionsManager.org
LICENCE          = GPL3
------------------------------------------------------------------------
EXECUTING HOST   = ws2.soho
------------------------------------------------------------------------
```

Part I

Common Basics

Chapter 1

Preface

1.1 History

Version	Date	Author	Description
01.03.003.a01[144]	2008.02.11	Arno-Can Uestuensoez	Initial pre-release as embedded printable help
01.07.001.a01[145]	2008.08.03	Arno-Can Uestuensoez	First major update with numerous additions and partial review.
01.07.001.b02[146]	2008.08.11	Arno-Can Uestuensoez	Minor editorial updates. A lot of tests, some fixes.
01.07.001.b03[147]	2008.08.12	Arno-Can Uestuensoez	Minor editorial updates.
01.07.001.b04[148]	2008.08.16	Arno-Can Uestuensoez	Enhancement of documentation and Web-Site.
01.07.001.b05a[149]	2009.01.26	Arno-Can Uestuensoez	Minor editorials.

01.03.003.a01/2008.02.11
: The first basic set of features including stack-aware recursive cancel for QEMU, XEN, and VMW. This release contains almost the whole set of first-time implementations(Section 4 'First-Time Implementations' on page 45) with personal copyright and offered license under GPL3.

01.07.001.a01/2008.08.03
: First major update with numerous additions and partial review. Additional HTML versions and subsets are generated. Still classified as preview quality, but may be stable enough for public production use, as personally applied.

01.07.001.b02/2008.08.11
: The first official stable release for flat-VM functions. The stacked VMs are present in first version, where the canonical syntax is implemented. Some more user friendly addresing with enhanced capability-recognition and validation will follow.

1.2 Contact

Public maintenance:	*acue@sf1_sourceforge.net*
Administrative contact:	*acue@UnifiedSessionsManager.org*
Commercial Services:	*i4p.com - Engineering Office Arno-Can Uestuensoez*

1.3 Some General Remarks

Some chapters still has to be completed, but are already present as a reminder, whereas the chapters identified as important are written first. The work is currently going on, thus reviews and comments are welcome.

1.4 Legal

All mentioned Red Hat products and their registered names are Trademarks of the Company Red Hat, Inc.

All mentioned SuSE products and their registered names are Trademarks of the Company Novell, Inc.

All mentioned VMware products and their registered names are Trademarks of the Company VMware, Inc.

All mentioned Google products and their registered names are Trademarks of the Google, Inc.

All mentioned Sun products and their registered names are Trademarks of the Sun Microsystems, Inc.

All mentioned Microsoft products and their registered names are Trademarks of the Company Microsoft, Inc.

All mentioned Intel products and their registered names are Trademarks of the Company Intel, Inc.

All mentioned AMD products and their registered names are Trademarks of the Company Advanced Micro Devices, Inc.

Xen is a trademark of XenSource Inc.

QEMU is a trademark of Fabrice Bellard.

i4p is a trademark of Ingenieurbuero Arno-Can Uestuensoez.

If some is forgotten, it will be added immediately.

1.5 Acknowledgements

And, of course, I want to thank VMware for supporting their excellent VMware-Server and VMware-Player for free. The VMware-Workstation product initiated to my mind a major step of change and inspired a lot how software is commonly used and developed.

Many Thanks to Mr. Fabrice Bellard for his QEMU, which is the only and one test base for me to demonstrate a nested stack of VMs and it's integrated addressing including state propagation algorithms for now.

Great thank to the inventors of Xen at the university of Cambridge. UK, for their efficient VM.

And, of course, I would have probably no chance without "googling", so, even though it has to do something with business, many thanks for bringing the information of the whole world - and as soon as contacted of the remaining universe for sure - to my desktop. Hopefully I cope the current amount before the remaining universe comes into the scene.

I am meanwhile an enthusiastic user of CentOS/RHEL and OpenBSD, so I am glad having the opportunity to express my thank this way to all supporting persons an companies. Particularly RedHat Inc. for their actual open minded distribution policy and the CentOS team for their great work, and the OpenBSD team for their ongoing support for a base of real security.

And, last but not least, I want to thank very, very much to all the countless contributors for the numerous excellent Open- and Free-Software I use. Hopefully I can express my commitment and thanks with this piece of software, and my next following projects.

And finally I would like to express my thank to my friend Dirk and his wife Gisela, for their patience and enduring support. Their support at all enabled me reaching this milestone, despite of all the various and countless challenges and throwbacks to be managed.

Arno-Can Uestuensoez
Munich, Germany
March 2008

Chapter 2

Abstract

The "UnifiedSessionsManager" with it's main component "ctys" - "Commutate To Your Sessions" - is a unified and simplified shell-interface for operations and management of local and remote sessions on physical and virtual machines.

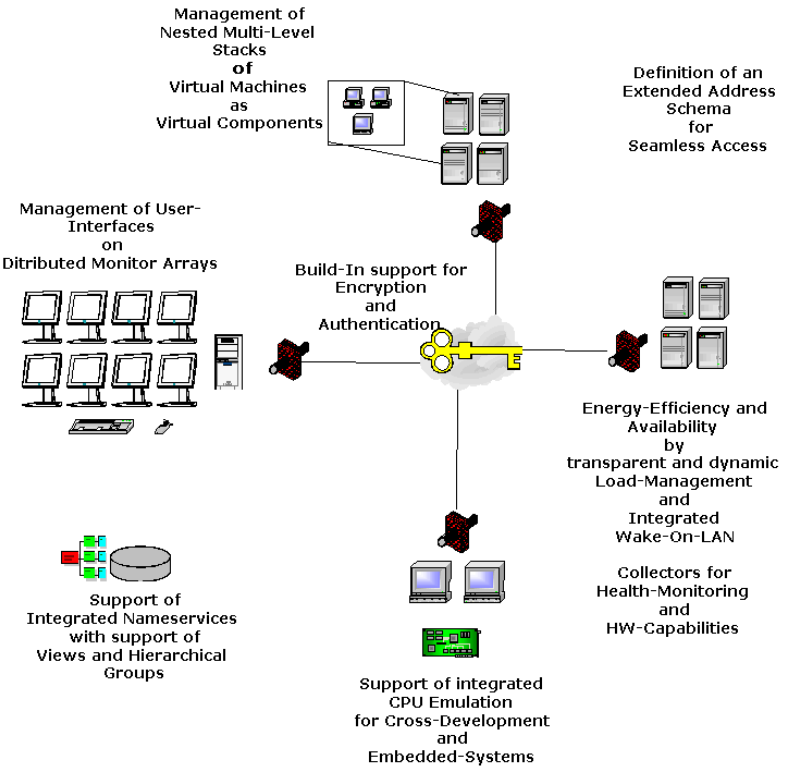

Figure 2.1: The UnifiedSessionsManager

The main features of the UnifiedSessionsManager comprise several first-time approaches combined with a simplified call-interface for technically complex setups. One of the most valuable assets is the identical call interface for all supported platforms and the transparently included mandatory security features,

ranging from a simple login up to Kerberos based distributed environments.

- Management of Nested Multi-Level Stacks of Virtual Machines as Virtual Components
- Management of User-Interfaces on Distributed Monitor Arrays
- Support of Energy-Efficiency and enhanced Availability by transparent and dynamic Load-Management and integrated Wake-On-LAN
- Poll systems information for offered HW-Capabilities and Health-Monitoring
- Support of Integrated CPU Emulation for Cross-Development and Embedded Systems
- Support of Integrated Nameservices with Views and Hierarchical Groups
- Seamless access to all types of sessions by the definition of an Extended Address Schema
- Support of encryption by SSH and authentication/authorisation based on one or more of the common approaches by SSH, Kerberos and SUDO.

Usability

The emphasis is clearly on the integrated and simplified usability of actually much more complex interfaces than the remaining visibility still contains. The main building block for this is the handling of the desktop presentation of the managed entities. This particularly comprises the advanced geometrical handling of session windows on an X11 based desktop with logically combined screens by the so called "Xinerama" mode. But also some smart addressing facilities for disconnection and re-establishment of sessions to headless-running server entities.

- Support of neatless logical addressing for multiple screens.
- Sub-positioning by screen aliases as customized in standard "/etc/X11/xorg.conf".
- Handles multiple "Screen Layouts" independent from the actually loaded layout.
- Supports for multiple desktops with a desktop aware job-scheduler for "flicker-avoidance" of intermixed calls for display on multiple desktops.

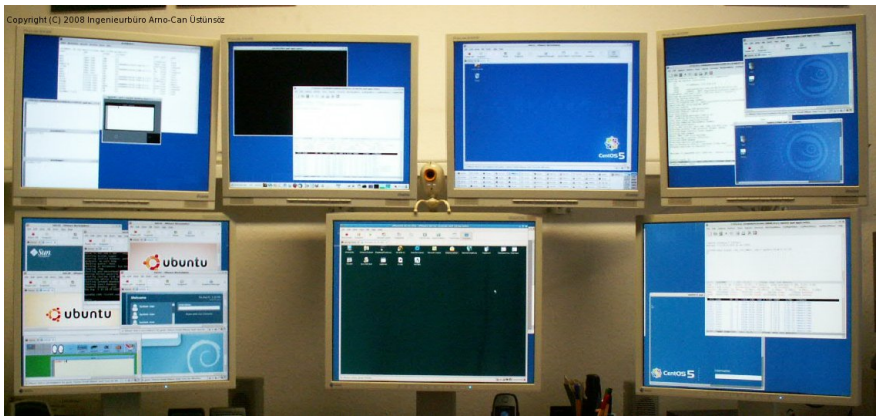

Figure 2.2: Physical Multi-Monitor Design

This screen layout contains(almost all entities are (TM)):

- 1x VMSTACK
- 4x CentOS, , 1x Fedora-8, 1x SuSE-9.3, 1x SuSE-10.2, 1x OpenSUSE-10.3, 1x 1x debian-4r3, Ubuntu-6.06.1, 1x Ubuntu-8.04, 1x OpenBSD-4.0, 1x OpenBSD-4.3, 1x Solaris-10, 1x MS-Windows2000

- 2x EMACSAM-Consoles
- 4x VNC-Consoles
- 5x X11-Consoles(here gnome-terminal)
- utilized by QEMU, XEN, and VMware
- Anyhow, this setup is far from the maximum frequently and easily utilized with the UnifiedSessionsManager.

Where all of them require different specific context-options due to presentation, security, and VM-Creation-Call requirements.

The benefit of the GROUPs and MACROs feature becomes quickly obvious, when the previous even simple example is shown by it's screenshot as resulting from the logical Xinerama-Screen.

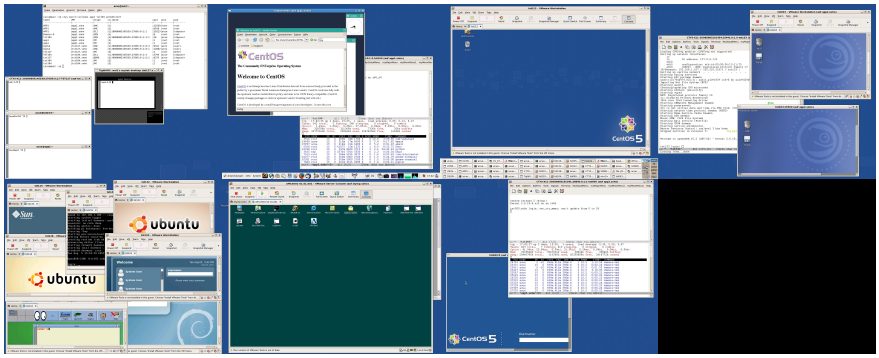

Figure 2.3: Logical Xinerama-Mode

The whole bunch of required calls could be pre-configured by MACROs and/or GROUPS with additional ordinary shell-facilities and for example could be reduced to the group "mydesktop". This includes the required boot of physical and virtual machines as well as the requested client for presentation and access on the local destop.

- ctys mydesktop

That's it.

Ah, yes, with the REUSE and/or RECONNECT, a simple repetitive call will re-arrange the whole desktop geometrically as well as by content, e.g. when one session is gone missing. If required, with

some simple bash-scripting each entity could be animated as a dynamic repositioning entity, making your windows-go-around.

And for sure, it should be mentioned, that the "chaining" of context-options combined with MACROs provides a simple means for multiple use of common generic GROUPs. Additionally available SUB-GROUPS offer support for different call-contexts within one GROUP. The include-feature provides modularized GROUPs, when bulk-starts has to be performed.

The termination of the group could be prepared even more simple, when the STACK-Propagation feature of CANCEL is utilized, thus resulting in a call like:

- ctys -t PM -a cancel myhostlist

Which by default performs a native and recursive shutdown of the whole set of stacked machines executed top-down.

Development Environment

The integration of Emacs and the foreseen integration of Eclipse provide additionally a neatless development environtment, where a multi-layered integrated debugging and testing approach could be applied to nested VMs.

- Guest-Application
- Guest-OS
- Hypervisor

Therefore as the first step Emacs is integrated as a Console-Type with one-layer and two-layered display variants.

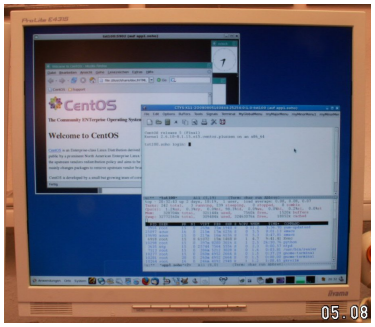

Figure 2.4: XEN Hypervisor/Emulator and CentOS Host within EMACS

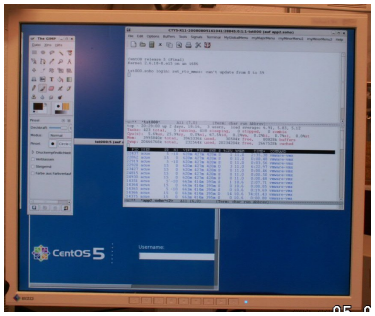

Figure 2.5: QEMU Hypervisor/Emulator and CentOS Host within EMACS

Resources

Yes, this amount of "actually and now" for download available features for the base system is probably the reason for a legthy abstract and required the number of the following facts, which are even not at all yet sufficient:

- Lines-Of-Code: more than 110.000
- Documentation pages: more than 600
- Required base tests for OS, and Hypervisors(let us say): 100 on at least 20-30 OS variants. Each test comprises the full setup and absolutely specific check of package variants for the specific hypervisor, HostingOS and the GuestOS. Frequently changing for various release versions.
- Required maintenance: definitely advanced, for example in case of involved VMSTACK entities and requested relocation capabilities.

Yes, the generic coverage of complexity has to be paid fully behind the scenes.

... and his work will go on.

Initial native support is provided for CLI , X11 , VNC , QEMU , VMW (VMware-Workstation/Server/Player), XEN , and PM (Linux and OpenBSD). Any OS is supported, when control by hypervisor only is sufficient.

Chapter 3

Feature Specification

3.1 Feature Introduction

The "UnifiedSessionsManager" introduces various features for the simplified usage of complex environments with multiple Operating Systems on distributed virtual and physical processing nodes.

The following features are some of the essential building blocks of distributed processing within semi-virtualized and fully-virtualized environments.

1. "Management of User-Interfaces on Distributed Monitor-Arrays"
   The user interface is based on X11 with support of Xinerama mode. Therefore an extended "geometry" is defined with embedded offset calculations and integrated support of logical screen addressing by user-defined labels. In addition support of "wmctrl" for multiple Desktops a.k.a. Workspaces is available.

   This version supports monitor arrays connected to one machine, the support for distributed Monitor-Arrays connected to multiple machines is now in lab and forseen for next version.

2. "Management of nested multi-level stacks of virtual machines"
   The stacking of physical and virtual machines with a arbitrary level is one of the basic design goals. It supports the implicit startup of required inactive parts of the VM-stack and the controlled shutdown of upper parts beginning from any level of a nested VM stack, including the handling of physcial machines as "stack-bottom".

3. "Energy-Efficiency and Enhanced Availability"

Basic load calculations by definiton of a cost based on nested stacks is defined, which offers particularly the analysis of the concrete active tasks with additional constraints.

Such constraints extend a pure resource calculation and adds a specific network service-aspect binding to a specific VM on a specific PM. For example a higher level VM may contain a Windows-Driver for a specific proprietary device, which is a mandatory prerequisite for other services and may run on one specific PM only, because it is physicall interconnected to that machine. Thus that VM has a joker for beeing kept available. The Wake-On-LAN facility is fully supported, for local segments, as well as for routed remote-segments. Various virtual bridge specific extensions and configurations are implemented.

4. "Collectors for Health-Monitoring"
   The "lm_sensors" and "hddtemp" projects are supported fully for each participating machine. Therefore any supported motherboard parameter, like fan speed, motherboard temperatures and voltages could be queried. In combination with the "hddtemp" utility the enumeration of temperatures(if HDDs support it) of all HDDs including 3Ware-Raid controllers is provided. Additionally "gkrellm" could be utilized.

5. "Collectors for HW-Capabilities"
   The detection of HVM, PVM, and PAE capabilities is included within the general platform attributes like type and number of CPUs, physical and virtual memory. Thus an inventroy information for the specific aspects of virtualization could be collected by a simple call.

   The same is supported for detection of logical capabilities, e.g. the installed hypervisors and their configuration, but limited for now to the current active kernel

6. "Support of Integrated CPU-Emulation"
   The emulation of various CPUs is supported by integration of the QEMU hypervisor. Therefore a new configuration file format based on bash is defined, which supports for pre-configured calls of QEMU by ctys.

   This feature is particularly of interest for Cross-Compilation and Embedded-Development. It could be be used in combination with the integrated support of Emacs "shell-mode" for

CONSOLE as an entry point for debugging with the "gdb-mode" or "ddd".

7. "Definition on an extended Address Schema"
   For the practical usability of a heterogeneous set of hypervisors a common addressing schema is defined, which includes the ability of handling configuration files by embedded search-and-match of several address attributes. These attributes could be combined for assembled attribute value assertions, when ambiguity occurs.

8. "Support of Integrated Nameservices"
   The essential building block for the management and seamless usage of various resources is the extended addressing schema in combination with an efficient name service. Therefore a flexible name service is defined, which offers cached acces for dramatic permance enhancements as well as access to offline machines when required for startup.

9. "Built-In support fo Encryption and Authentication"
   The design of ctys is based on assurance of network security from the first step on. Therefore the whole communication relies on OpenSSH only. The only type of connections supported are encrypted connections, within the local LAN as well as connections spanning external networks. Particularly support for kerberized SSH connections is build in, even though any authentication facility of OpenSSH could be utilized. Additional support for "ksu" and "sudo" based access permissions is build into the core functionality for restricted systems resources.

Some additional essentials which should be mentioned here are:

- Introduction of "GROUPS"
  Any set of execution targets, including context specific options, could be combined to an addressable new object and stored within a simple ASC-II file. Hierarchical includes of nested and chained groups are supported.

- MACRO feature for arbitrary CLI positions
  Any part of the CLI - except the "master" command itself - could be defined as a MACRO, which itself could be a hierarchical defined collection of MACROs.

- Full customizable generic tables "TAB_GEN"
  Any data output could could be filtered by generic tables, which could be arbitrary defined by the user. Any position, size,

and field-clipping, including multiple displays of the same field, could be defined, and optionally stored as a MACRO.

Common basic conversion between IP and DNS representation of TCP addresses is included.

REMARK: In this version the generic table filter is supported by ctys for LIST and ENUMERATE, and by the support tool ctys-vhost, all others will follow a.s.a.p. Please verify "all others" versus this statement whether document has already to be updated.

- "Connection-Forwarding and Display-Forwarding"
  The distinction and full support of local clients in combination with remote servers by splitted acces, as well as coallocated operations with X11 based display forwarding only is fully supported. Therefore in case of splitted processes an encrypted so called "tunnel" is created and utilized implicity by usage of the OpenSSH feature for port-forwarding. All actual present ctys-tunnels including the information on interconnected ports are displayed by LIST call on the callers machine.

  Thus the so called "headless" mode is supported for any appropriately configured hypervisor and VNC. This is for obvious reasons not available for CLI and X11 plugins.

- "Extendibility"
  The architecture of ctys is targeted for open and easy extension by writing plugins as bash modules. Therefore ctys defines - and is implemented completely based on - dynamic loadable and self- configuring bash-plugins, which serve as thoroughly documented patterns for specific extensions. The help call of ctys supports for various listings of the internal functio interfaces.

3.2 Feature-Sum-Up

The listed PC, Workstation and Server based platforms with listed Hypervisors are supported and tested when marked with "OK". Ad-

ditional platforms are going to be added for next versions("*").

The PMs support as a "virtual but Physical Machine" is divided into the bottom-level and upper-level PMs. The bottom-level PM could be seen as the "controlling lowest hypervisor" layer, which supports actual "close to real world HW" functionality. The upper-layers beginning with level=1, implement a virtual HW environment, which is the "as seen physical HW" for their contained hypervisors. The PMs in general host the HOSTs plugins as native GuestOS access functionality.

The utility "ctys-genmconf" supports the detection and generation of relevant control data, the utility "ctys-plugins" verifies actual available operational states and resulting "horizontal" and "horizontal" features. Both tools could be applied locally as well as remote for multiple targets, but work for now sequentially. Thus the bulk-application has to be provided by the user, same for the tool "ctys-setupVDE"

The main development and production platform for the Unified-SessionsManager is CentOS(beginning with 4.4), thus the Unified-SessionsManager should be fully compatible to RedHat(TM)-Linux, which is planned to be supported continuously. Compatibility with any derived distribution like Oracle(TM) "Unbreakable Linux" and "Scientific Linux" should be given from-the-box and is planned to be verified for the "main distros" within one of next versions. The former base was SuSE(TM)(generally beginning with 1.x/1993(?)).

The main base-OS for networking devices is OpenBSD(beginning with 2.x), thus this is planned to be supported continuously.

The integration with the project "zamklibant" [151, ZAMKLIBANT] is foreseen. "Zamklibant" is a general purpose OS and VM builder, foreseen as a building block for "v-components", scaling from Main-Servers to tiny-devices. "Zamklibant" is planned to be released following the "UnifiedSessionsManager".

It is expected for the future, that virtual components - "v-components" - will extend the design of modularity by Software-Components with a higher level of encapsulation and flexibility than current available technologies.

Supported OS-Distribution vs. Containing Plugins.

Distribution[1]	PMs		VMs					
	0	1+n	KVM	VMW	XEN	QEMU		
	x86	x86	x86	x86	x86	x86	ARM	Coldfire
CentOS-5.0	OK	OK	-	OK	OK	OK	-	-
CentOS-5.1	OK	OK	-	OK	-	-	-	-
CentOS-5.2	asap	asap	asap	asap	asap	asap	-	-
Debian-4.0r3	OK	OK	*	OK	*	OK	(OK[2])	-
eCos	-	*	-	-	-	-	*	-
Fedora-8	OK	OK	*	OK	OK	OK[3]	-	-
FreeBSD-7.0	*	*	*	*	*	*	-	-
MS-Windows[4]	-	-	*	OK	*	(*)	-	-
NetBSD-4.0	*	*	*	*	*	*	*	*
OpenBSD-4.0	OK[5]	OK	-	OK	-	OK	-	-
OpenBSD-4.3	OK[5]	OK	-	OK	-	OK	*	-
OpenSolaris	*	*	-	-	*	*	-	-
Solaris-10[6]	(OK)[5]	(OK)		(OK)	-	*	-	-
SuSE-9.3	-	OK	-	OK	-	-	-	-
SuSE-10.2	-	OK	-	-	OK	-	-	-
openSUSE-10.3	OK	OK	-	OK	*	OK	-	-
Ubuntu-6.06.1	*	OK	-	OK[7]	-	-	-	-
Ubuntu-8.04	*	OK	-	OK	*	*	-	-
uCLinux	-	*	-	-	-	*	*	*

Table 3.1: Tested OS support for Stacked-VMs on PC-Platform

[1] Contained distribution. For "containing" distributions, founding the execution base of VMs refer to the Table:3.3

[2] Currently raw only, shifted due to required build from scratch.

[3] Some synchronity problems with localclock and/or ntpd, performance problems leading to hit of timeouts for native HOSTs VNC.

[4] Control by hypervisor only, no native support. Cygwin is foreseen for eventual future adaption. Tested with several versions, e.g. Windows-NT-Server, Windows-2000, and Windows-XP.

[5] Current version without WoL for itself, but supports wakeup of others. Supported Plugins: HOSTs and PM.

[6] Some severe limitations may occur for Solaris, due the limitation of the "args" output of "ps" command to 80 characters. Thus the LIST action is faulty for some plugins, which means the instances are simply hidden due to argument-parts truncated by "ps". Some specific adaptations will follow. This depends on the argument ordering of the current command/wrapper and the actual contents beeing truncated. Supported Plugins: HOSTs and PM.

Supported Hypervisors.

Hypervisor	Versions
KVM	ffs.
QEMU	0.9.0, 0.9.1
VMware-Player	1.0.4, 1.0.5
VMware-Server	1.0.4, 1.0.6
VMware-Workstation	6.0.2, 6.0.4
Xen	3.0.3, 3.1.0

Table 3.2: Tested Hypervisor Versions

Supported native plugins vs. OS-Distribution.

Distribution[8]	PMs	VMs			HOSTs		
	PM	QEMU[9]	VMW	XEN	CLI	VNC	X11
CentOS-5.0	OK	OK	OK	OK	OK	OK	OK
CentOS-5.1[10]	OK	OK	X	OK	OK	OK	OK
CentOS-5.2[11]	asap	asap	asap	asap	asap	asap	asap
Debian-4.0r3	OK	OK	*	*	OK	OK	OK
eCos	*	-	-	-	*	-	-
Fedora-8	OK	OK	*	OK	OK	OK	OK
FreeBSD-7.0[12]	*	*	*	*	*	*	*
MS-Windows	-	-	-	-	*[13]	*[13]	*[13]
NetBSD-4.0[14]	*	*	-	*	*	*	*
OpenBSD-4.0	OK	*	-	-	OK	OK	OK
OpenBSD-4.3[15]	OK	*	-	-	OK	OK	OK
OpenSolaris	*	*	-	*	*	*	*
Solaris-10[16]	(OK)	-	-	-	(OK)	(OK)	(OK)
SuSE-9.3	OK	*	-	-	OK	OK	OK
SuSE-10.2	OK	*	-	-	OK	OK	OK
openSUSE-10.3	OK	OK	OK	*	OK	OK	OK
Ubuntu-6.06.1	*	*	-	*	OK	OK	OK
Ubuntu-8.04	*	*	*	*	OK	OK	OK
uCLinux	*	-	-	-	*	*	*

Table 3.3: Tested Native Plugins vs. OS-Distribution

[8] Containing distribution, which is capable of executing an upper VM within the nested VM-Stack, if marked so by a "VMs".

[9] Reminder: QEMU is the currently available building-block for VM-Stacks, thus any "intermediate-VM" requires the native execution of a QEMU-VM(see figure 8.1).

[10] Should work from-the-box, shift is just a question of priorities.

[11] Should almost work from-the-box, shift is just a question of priorities. Probable minor challanges with the "Xen-3.x" version, but are not really expected.

[12] Minor adaptions expected.

[13] Support eventually by cygwin.

[14] Minor adaptions expected.

[15] Should work from-the-box, shift is just a question of priorities, same for any OpenBSD-4.x release.

[16] Some severe limitations may occur for Solaris, due the limitation of the "args" output of "ps" command to 80 characters. Thus the LIST action is faulty for some plugins, which means the instances are simply hidden due to item-parts truncated by "ps". This depends of the argument "arbitrarily" re-ordering of the actually utilized command/wrapper and the resulting contents actual beeing truncated. Supported Plugins: HOSTs and PM.

Future plans.

Feature	HostOS	GuestOS
Update CentOS to version 5.2.	N	N
Integration of KVM - Kernel based VM.	N	N
Full EMACs as IDE integration with it's own plugin.	N	N
Full gdb/ddd integration	N	N
Virtual encrypted circuits	N	N
Nagios Integration	N	N
SNMP support	N	N
LDAP support	N	N
Remote RS232-Console Server	N	N
VM relocations/live-migration	N	N
UPS Integration	N	N
Integration of IP-Console	N	N
Shift or Migrate between different instances of the same hypervisor.	N	N
Evaluate and probably integrate OpenVZ.	N	N
Evaluate and probably integrate VirtualBox.	N	N
Shift or Migrate between different hypervisors		
Integration of remote Power-Switches		
Distibuted Screen Arrays attached to multiple workstations	N	N
Touch-screen support		
IPsec support		
RADIUS support		
Eclipse integration		
Cygwin support on Windows(TM) platforms		
Extension to IPv6		

Table 3.4: Planned roadmap for new features(N:next vers.)

Chapter 4

First-Time Implementations

This software is donated to the public based on GPL3, the first public release is dated on January 2008.

The terms of GPL3 particularly comprise the concepts and the design principles for the project "Unified Sessions Manager" as listed here non comprehensively.

4.1 First set - 2008.02.11

VM-Stacks
: The basic building block for stacked VM handling, including some of the advanced multi-ISO-Layer-address-handling and the derived technologies within this software.

  The whole theory and technology, as well as the concepts of the designed and implemented, and as upcoming described feature previews.

Management of nested stacks of Virtual-Machines
: The concepts, design, and implementation of the escalation of dependent and remapped state change actions for virtual and/or physical machines, when additional single or nested virtual instances are operational. The state change propagation by remapping to appropriate states when propagating into upper layer.

Appliances as ordinary SW-Components
: The concepts, design, and implementation of the usage of a

huge amount of virtual machines on bulk-core-CPUs with hundreds or even thousands of cores, where due to expected enhancements of processing power this seems to be used in a very ordinary manner.

Thus the potential is even the replacement or better extension of ordinary system processes by virtual appliances. Offering a much more flexible design and operational base for network relocation, component based availability enhancements, and encapsulation.

Integrated self-reconfiguring Network Management Interfaces

The "Integrated Management Interface for self-reconfiguring Network Management Systems applied to VM-Stacks" contains the concepts, design, and implementation of the usage of an attached functionality to ordinary graphical icons, but positioned on the screen in order to represent the physical or logical structure of a networked environment for VM-Stacks. Therefore an machine interface is defined, as a case study based on Nagios, where a via CLI started managed entity, which could be a virtual machine and/or a physical machine, registers itself by usage of a specific differentiated icon as a dynamically attached entity.

This will provide multiple views, particularly a view representing the nested containment-like execution stack in various views itself. Therefore a tree-view, a nested boxed-view, and a staple of bricks-view is defined.

Independently from this additional VM-Stack-View, any networked entity is represented by it's secondary logical nature as an ordinary networked device, which transparently covers the primary characteristic as a virtual entity.

Load distribution of stacked virtual machines

The concepts, design, and implementation of the usage of an system in order of evaluation COST values for load distribution within nested entities.

Even though the overall CPU could be measured by monitoring the lowest container within the stack, this is not necessarily

true for management of resources accessed by software applications from within virtual entities on higher levels of the stack only. This is frequently true, when a common network platform e.g. by Linux OS is defined with additional virtual instances as "worker-entities" only. When e.g. scanners, printers, or other devices are supported as a fabrication group with load distribution, the overall load of the stack base has no relevance to the FIFO character of the required resources. Thus any stacked element could have it's own constraints of several types, which is not visible outside the entity itself. This particularly could appear when e.g. embedded systems are simulated by usage QEMU for CPU simulation and execution uCLinux, whereas the QEMU instance itself is executed within a so called DomU of Xen.

Performance Enhancements for address resolution

The applied technologies for replacement of highly sophisticated address resolution by so called Attribute-Value-Assertion as commonly used within CMISE/CMIP/Q3 and SNMP environments for structured access to data, are claimed to be independently invented by myself and donated to the public by GPL3 too.

This technology comprises the pre-assembled generation of specific table entries with seperated fields which are handled as a single entity for cascaded application of regular expression based simple matching filters. This pattern-matching on flat records lead to same data-results as huge ASN.1 based approaches, but does require a minimum of effort.

The advance uprises from the combined handling of the overall record for flat-matching by regexpr and the read-out of data based on structured records by fields. Particularly the opportunity of chained filter application on the intermediate sets of results leads to quite good matching results with more than satisfying overall performance.

The practical application advantage is that the almost "trivial" framework of common UNIX base-tools are required only and the average access time still remains within milliseconds even when using bulk PC components and implementing it as part of this software, by a simple awk-script.

Component-Oriented bash usage
The consequent utilization of components as dynamically loaded components similar to "shared libraries/objects" by "sourcing" seems to me to be a new first-time approach.

Arno-Can Uestuensoez
Munich, Germany
December 2007
(2009.01 minor editorials)

4.2 Second set - 2008.07.10

Vertical-Stack-Operations
Maping of each task affecting a single "row" of Stack-Operations targeting a unique VM as a destination into it's own execution context by technically defining a specific controller process (see Section 13.3 'Hosts, Groups, VMStacks and Sub-Tasks' on page 161).

The controller may keep control as a classical CONTROLLER and distribute any task related subtask by itself, or it may forward the responsibility in a NESTED manner for each next step recursivley to an instance natively executed within the current "uppermost" stack instance.

Grouped-Stack-Operations
In advance of mapping each single "row" of Stack-Operations into one task, Wildcards in - recognition of vertical depency - may be applied, in order to support an expansion for specific levels. The expansion of the set and/or wildcard on each Stack-Level must guarantee the availability of the founding peer for each entitiy to be activated within the next layer.

Stack-Capabiliy-Evaluation
The current approach is targeting a heterogenous set of hypervisors to be supported by a single and almost unique interface. Therefore a common set of operations in a minimalistic-approach, called ACTIONs is defined, which almost provide an identical syntax.

Particularly the intermixed usage of various combinations within a single VM-Stack requires rises some compatibility issues to be considered by the system tools. An approach of demanding

the user to interact on it's own absolutely semantically, correct including recognition of actual resource exhausts seems not to be appropriate.

Thus within an cache database - th cacheDB - as the central knowledge base, several attributes are defined, in order to assure an automated static and dynamic verification of the call-compatibility including the resource-availability-compatibility of stacked VMs. The most important attributes are:

STACKCAP
: The offered stack capability for upper entities.

STACKREQ
: The required stack capability for execution base entities.

HWCAP
: The offered either physical or virtual hardware capability for upper entities.

HWREQ
: The required either physical or virtual

Each of this attribute will be in addition available as a dynamic runtime variant assembled initially during each startup. The contained subrntries are assemblies of various specific attributes, representing partial capabilities for various tasks. Thus the more or less static execution verification is supported as well as some sophisticated distribution algorithms. This include the dynamic variation of startup-assigment of available resources such as the Virtual-RAM and the number of Virtual-CPUs in case of VMs, as well as detection of specific HW-requirements, e.g. in case of local-only available scanner or ICE for embedded development. Almost any pre-requisite could be customized and handeled by the implemented code.

Even though this requirement is expected to be identified and solved before, the consequent application of stacked VMs is as far as known a first time approach described originally within this document.

A secondary, not less important application is the reloaction of active VMs, even between different hypervisors/VMs. This

frequently requires some specific hardware and of course hypervisors to be available. The presented and implemented approach even supports a fine-grained definition and recognition of version and subversion definitions for each component. This could be within the available implementation easily customized.

Related technical process-modells as first-time-invented could be reviewed, tested, and applied by the supplied code under GPL3 license.

Virtual-Circuits

Virtual-Circuits is seen within the UnifiedSessionsManager as a neatless integrated, to say inherent, facility in order to establish a typical relay based peer-to-peer communications line. This is performed in a multi-layer approach with an top-level peer-to-peer encryption assuring the exclusion of intrusion capability on any intermediate section-relay.

This approach seems not to be new, as shows a similar for example presented by the very impressive overall approach on [131, VIRTUALSQUARE].

But the independently developed degree of integration into an actually available utility comprising a concept for handling almost any aspect of user sessions, with various connection and Client/Server location types, is - even though still far from beeing perfect - the first time approach as far as known.

Arno-Can Uestuensoez
Munich, Germany
July 2008

Chapter 5

Secure Sessions

The basic idea behind ctys is to support a common access framework for a combined environment

- intermixed with multiple OS(Unix, Windows)
- running on distributed and intermixed
- physical and virtual platforms.

Therefore some common software plugins like VNC, X11, XEN, SSH, Kerberos, LDAP and automount are combined together by usage of ctys in order to supply simplified creation and transparent access to sessions.

The implementation of ctys as a "bash" based shell script provides flexibility when using it within login-profiles, for batch files, and interactive from the desktop.

The management of the sessions client windows on X11 based desktops is supported broadly, as probably one of the most important task for distributed environments. Therefore the X11-geometry option is extended based on the standard configuration file "xorg.conf". Any position related section, including multiple "LayoutSections" could be addressed symbolical or numerical. Particularly in Xinerama mode the required offsets for windows positions are calculated as required, also when symbolic Screen-Names are used for extendedGeometry. The resolutions of servers and clients could be set seperately within one call, when supported by the sessions base application like VNC. In addition the addressing means of "wmctrl" for multiple desktops are supported when the tool is detected.

The titles of the windows, which are visible on the desktop and as labels within the taskbar, could be used in a syntactically unified manner for addressing any ctys managed session.

The resulting calls could be particularly for pre-set default values quite simple:

```
ctys -a CREATE=label:COSOLE,REUSE host01
```

This line first checks on the host01 whether a VNC session with label "CONSOLE" is already running. If so, a server-local vncviewer is started and connected to that server-process. The whole connection, including the display forwarding is established through an SSH tunnel. When no session with given label exists, a new one is started and connected to a vncviewer.

Next example starts a VMware-Workstation/Server/Player with native gui on the host "host01".

```
ctys -t vmw  -a CREATE=fname:'vmware/openbsd-001.vmx',\
       REUSE host01
```

or

```
ctys -t vmw  -a CREATE=id:'vmware/openbsd-001.vmx',\
        REUSE host01
```

or

```
ctys -t vmw  -a CREATE=label:OpenBSD-001,REUSE  host01
```

Last three examples are executed with the identical vmx-file, if "label:OpenBSD-001" matches "displayName" within "fname:vmware/openbsd-001.vmx", which will be searched by "find" command. The VM is immediately opened and executed due to default call options "-x -q -l". The GUI is closed when VM is shutdown. The vmx-file is located relative to caller's remote HOME. The default mode is "-L DISPLAYFORWARDING", which starts a coallocated client on the same machine as the server component is executed.

One of the basic advantage is the comprehensive unification of addresses for sessions with the introduction of a common addressing layer. This assures a common means of direct or indirect user defined LABELS as an alias for usage as sessions identifier. The mapping will be done completely dynamic, so no persistent extra database is required. Mapping for VMs is based on their various "name-entries",

e.g. the Domain-Name or the Display-Name. The LABEL is forced to be visible on several places, so it is used as call option, windows title, or if no otherwise possible as a masked comment on CLI, still visible with ps-command. Due to usage as window title it is visible in the taskbars of most destop managers.

These LABELs will be used for the "ssh-tunnel" call too, so a simple ps will show which "tunnel" is related to which session, displaying it's ports and various sessions parameters for "-L CONNECTION-FORWARDING".

In addition some handy extensions are available, like using UUIDs, MAC-addresses, TCP/IP-addresses for addressing of VMs. This could be used by defining path prefixes for "find and scan" options, which even allows the move of VMs without change of addressing.

Be aware, that security could be seen as almost perfect for network packages, but lower, once local access is provided. Same as for Kerberos e.g.

5.1 Security and Authorization

Ctys itself utilizes SSH with port forwarding in various manner, though utilizes the security mechanisms of SSH and additional components. For the deeper understanding of the concepts and facilities used within the UnifiedSessionsManager it is strongly recommended to read the books *"SSH The Secure Shell"*[26, SSHDefGuide] and *"Kerberos - The Definitive Guide"*[25, Krb5DefGuide] first.

In addition for usage of LDAP fo user information *"LDAP - System Administration"*[29, LDAPadm] and *"LDAP verstehen, OpenLDAP einsetzen"*[31, LDAPverst](german only) should be referred to. When using LDAP with certificates *"Network Security with OpenSSL"*[27, SSLDefGuide] is helpful.

For general security issues the books *"Virtual Private Networks"*[33, VPN], *"SELINUX - NSA's Open Source Security Enhanced Linux"*[34, SELinux], *"LINUX Security Cookbook"*[32, LinuxSecCookbook], *"Secure Architectures with OpenBSD"*[14, PALMER&MAZARIO], and *"Absolute OpenBSD - UNIX for the Practical Paranoid"*[15, LUCAS] are recommended.

The actual advance could be gained, when any kind of pre-configured direct or indirect network login is supported. In such a case - a.k.a. Single-Sign-On or SSO - supported by the underlying systems, no additional user interaction is required for authorization.

If a network account is not available, you have to type your password by means of SSH each time you access a remote account, which could be somewhat threatening, when you use a bulk feature for tests or automatically open several remote sessions during execution of your login scripts. Same for connection-forwarding , where multiple password requests occur for pre-fetched
remote-port information and a pre-established tunnel before finally connecting the client to the local forwarded port. Thus, most benefit is given when any kind of an SSO is provided, particularly with kerberos and SSH, probably in combination with LDAP and an LDAP based automount.

Another point to underline here is the "-b" option, which internally leads to a call of ssh with it's "-f" option. The "-f" option of ssh handles forking of processes to background operation after required authorisation dialog, such as password entry when no SSO is provided. So, the "-b" option forces the ssh-client into background operation for immediate release of the current console.

Do not use "&" instead.

For some display functions such as ENUMERATION or LIST synchronous operations is default and might be preferred.

Superposing and extending the synchronity is the parallel execution of remote tasks. For additional information refer to "-b" option.

5.2 Xinerama Screen Layouts

The most of the following examples are based on the monitor array described here. Each screen is configured as 1280x1024, with the following ServerLayout. The Identifier="Layout[all]" is as default of Gnome/Xorg, but any number of specified ServerLayouts as supported by X11 is applicable and could be addressed of course. Any number of Screens in any combination and mapping, as well as related indexing is supported.

REMARK: 1. There is some minor limitation within this ver-

sion. This is due to the calculations of screen positions for logical addressing screens by their numbers or labels - see "-g" option of ctys.

The keywords for configuration of X11 - such as "leftof" - are not supported within the so called "geometryEx" internal position calculations for logical screen addressing.

Therefore the absolute screen positions as numerical values have to be used within xorg.conf. This is e.g. the default for CentOS based distributions and current versions of NVidia utilities.

2. The usage of "-" as minus for screen positions of geometry is not supported in this version.
   This seems not to be a real thread, because the supported platforms have positive absolute positions only.
   Relative addressing from a viewpoint other than the standard (0,0) is not supported.

In the following sections describe the reference ServerLayout for folowing tests.

5.2.1 Physical Layout-1

This example is a physical layout with several low-cost dual-port graphic cards, all screens are attached to the same computer. The X11 operations mode is Xinerama.

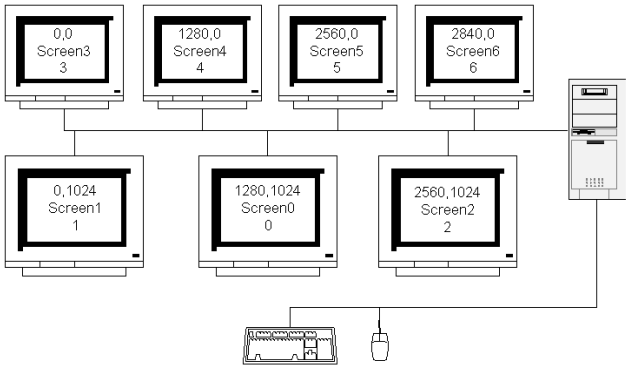

Figure 5.1: Physical Multi-Monitor Layout-1

The actual hardware design shows an two-row layout, where the upper screens are foreseen for monitoring tasks and temporary storage of working sessions, whereas the lower row is the actual working area for development and testing tasks. Thus the upper screens are 17'-monitors, whereas the lower are of 19'-inch.

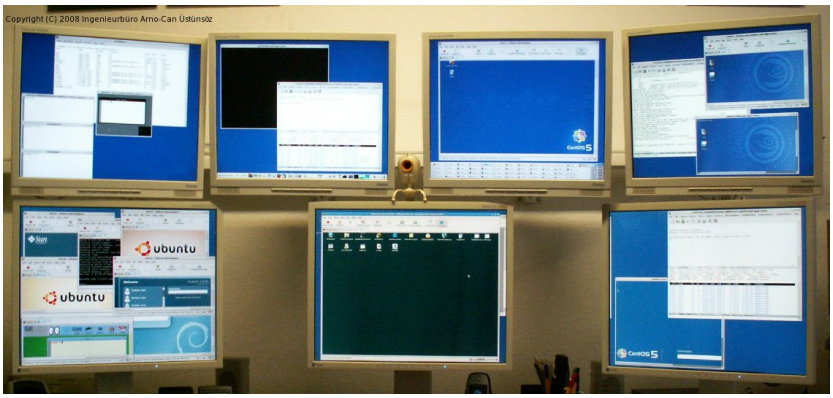

Figure 5.2: Physical Multi-Monitor Design

Logical Layout-1a

The logical representation of the previous physical layout within "/etc/X11/xorg.conf" is as follows. The X11 operations mode is Xinerama.

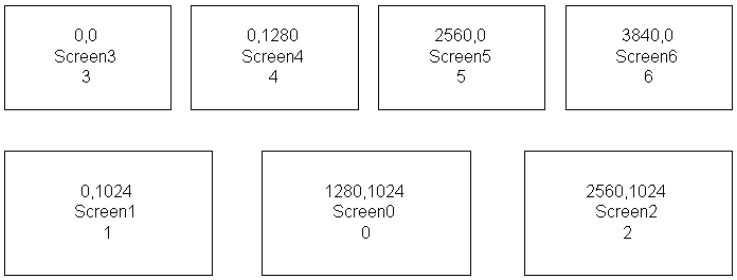

Figure 5.3: Logical Multi-Screen Layout-1

The Xinerama mode represents all involved screens a one logical screen which could be seamless accessed. Therefored the display size of the whole screen founding the "big-logical" is the sum of the involved screens. The following screen-shot shows the previous

example from Figure 5.2 as screenshot, which comprises all partial screens.

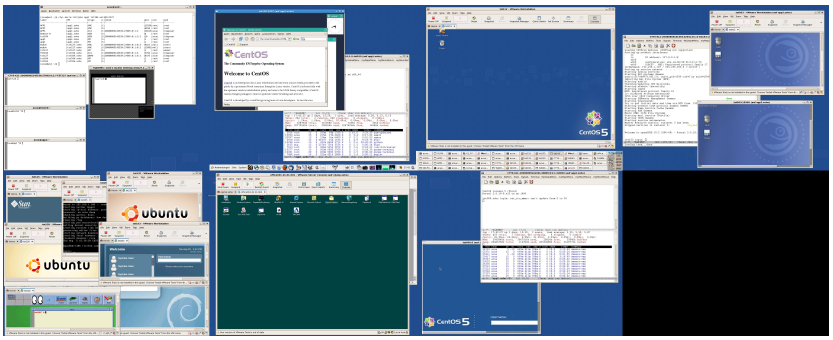

Figure 5.4: Logical Xinerama Layout

In the current design it was choosen to configure each screen with the same size of 1280x1048 and accept a "whole" in the bottom-row.

The Screen numbering here was the original positioning on the first motherboard on the used workstation with 1xAGP and multiple PCI graphichs cards. Each of which had 2 Ports and GPUs, some were actually connected to one monitor only.

Based on this the cabling was designed and numbered, and of course installed in the "cable-tree". These more or less fixes the connectors in the "bulk" of the cabling bundle.

Logical Layout-1b

The logical representation of the previous physical layout within "/etc/X11/xorg.conf" changed as follows after required changing of the motherboard. This is due to the fact, that the new motherboard has 2 PCIe and 4 PCI slots, whereas the old had 1 AGP and 5 PCI slots. The slots are intermixed by their positions due to the assumption of the manufacturer for utilization some SLI cards with oversized thickness. Anyhow, due to less importance of graphics processing itself only fanless and low-power standard GPUs with standard sizes are used.

The numbering order of the busses changed due to the different interconnection of chipset and it's busses and the physical positions of it's slots, though the plugin positions. Thus the physical positions

of the logical X11 screen names changed too.

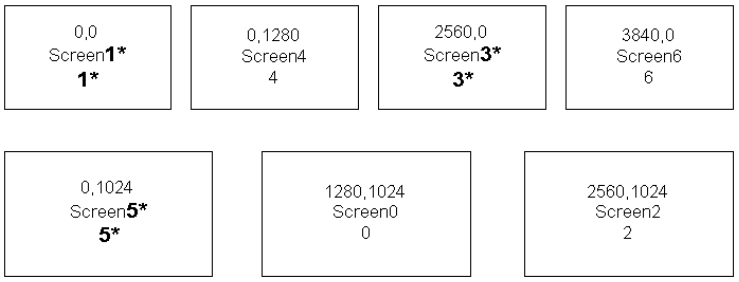

Figure 5.5: Logical Multi-Screen X11-Remapping

So due to logical remapping of the screen positions it was not neccessary to reposition the cables, but just to redefine some labels.

By manual configuration of names within "/etc/X11/xorg.conf" the above array could be prepared for logical addressing.

One possible naming could be an 2-dimensional array simulating Index-Style.

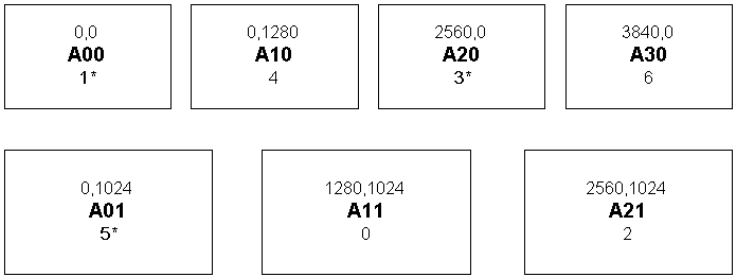

Figure 5.6: Logical Multi-Screen X11-Array-Style

The final physical monitor array with it's logical addressing is:

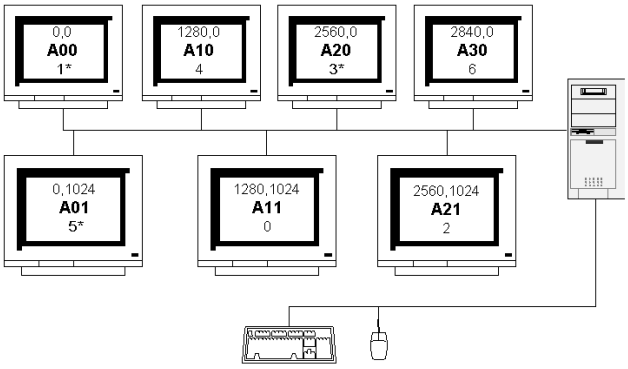

Figure 5.7: Physical Multi-Monitor Array-Style Addressing

Now the physical "Screen3", which became the logical "Screen1*", could be addressed as "A00", the "Screen5*" as "A01".

label	physical	logical
A00	Screen3	1*
A10	Screen4	4
A20	Screen5	3*
A20	Screen6	6
A01	Screen3	5*
A11	Screen4	0
A21	Screen5	2

Table 5.1: Mapping schema of labels to screens

These symbolic names could be literally used within ctys the "-g" options and it's value <geometryExt>.

Logical Layout-1c

In addition to addressing physical screens and positions, the full scope of GNOME desktops is supported.

The workspace feature is utilized by usage of the tool "wmctrl". Thus a fully qualified addressing is provided by encapsulating the whole set of tools within a seamless view.

The following functionality is only available when "wmctrl" is installed appropriately, else the optional path element of workspace is just ignored and the current visible workspace is used.

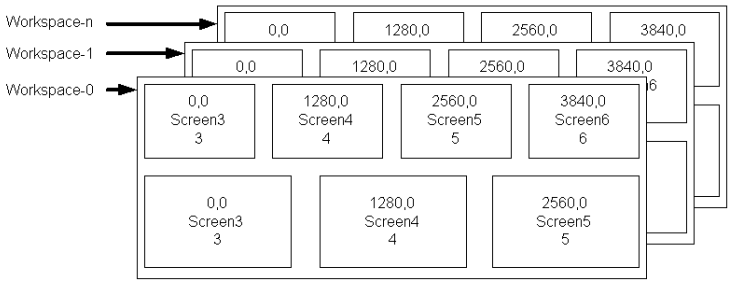

Figure 5.8: Mapping schema for multiple Desktops/Workspaces

Desktops/Workspaces could be selected by their numerical index or literally by the user configured string-representation.

Within the user supported labels of Desktops/Workspaces SPACES are not supported, and the usage of SPECIAL CHARACTERS should be avoided, else the numerical ID could be used only.

5.2.2 Physical Layout-2

This example is a physical layout with several low-cost dual-port graphic cards, which are attached to multiple computers. In addition two touch screens are attached to the first machine. The touch screens are supporting complex context specific User Interfaces for quick decisions of applications operators. The size of the monitor only screens are given as 1280x1024, whereas the sizes of the touch screen is Screen8(1048x768). The Number of the screens is just limited by processing capacity.

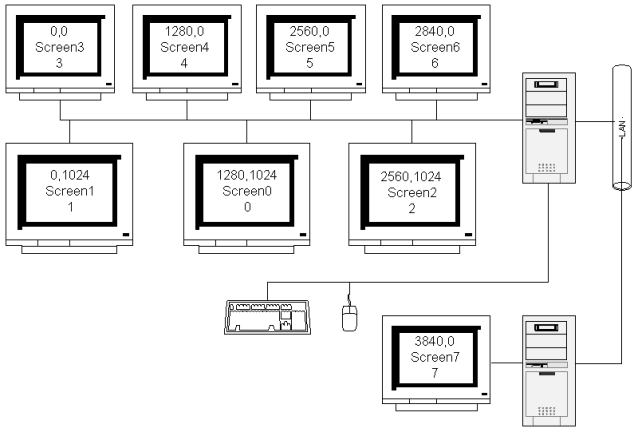

Figure 5.9: Physical Multi-Monitor Layout-2

This configuration is in current version supported by means of the package Xdmx, which has to be pre-configured. Based on this configuration the virtual screen - e.g. in Xinerama-Mode - will be managed by ctys.

5.3 Client-Session Windows

This section handles primarily the functionality related to visual layout on the desktop. Even though the type of session involves in cases of a virtual machine the startup and/or connection to that, this will be focused in the next chapter, the visual aspects are reviewed here.

So the following basic configuration is the standard case to be handled. It represents a close peer-to-peer relation of a client initiating the session and a server which hosts the applications a.k.a. XClients. In this case the Client only serves as a thin client which just executes the vncviewer. Two standard cases are supported. First, the Xserver and Xclients for the application are all together located on the server. Second, the Xserver located on the server, whereas the Xclients is started on the client. In both cases the communications is performed port-forwarding feature of ssh.

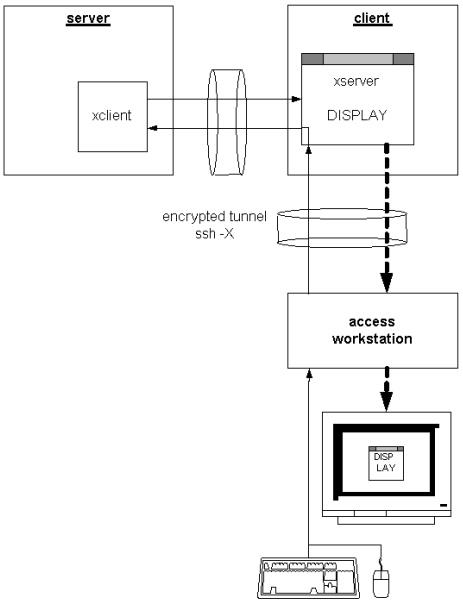

Figure 5.10: Basic handling of client sessions windows

Setting up a layout based on xorg.conf-entries will be done for example by the following call:

```
ctys -t vmw -a CREATE='f:vmware/openbsd-001.vmx' \
-g "600x400+2660+100" host01
```

This Starts a VMware-Workstation/Server session on a Xinerama group with offsets and size given by "-g" parameter.

The given values yield to a GUI-Window on "Screen5" with Index=5 which is in the offset range of +2600+100. The upper left corner has the coordinates (2560,0), thus window is positioned with an relative offset of (100,100)=+100+100. Starts a VMware-Server session on a Xinerama group with offsets and size given by "-g" parameter. The given values yield to a GUI-Window on "Screen5" with Index=5. The upper left corner has the coordinates (2560,0), thus window is positioned with an relative offset of (100,100)=+100+100. Same could be now given in one of the following ways:

FullyQualified Screen-Addressing by labels from xorg.conf.

```
ctys -t vmw -a CREATE='f:vmware/openbsd-001.vmx' \
-b on -g "600x400+100+100:Screen5:ServerLayout" host01
```

Qualified Screen-Addressing by labels from xorg.conf, using the default for missing ServerLayout, which is defined as "take the first section".

```
ctys -t vmw -a CREATE='f:vmware/openbsd-001.vmx' \
-b on -g "600x400+100+100:Screen5"  host01
```

Same as before, but using the numerical index of the screen instead of it's label.

```
ctys -t vmw -a CREATE='f:vmware/openbsd-001.vmx' \
-b on -g "600x400+100+100:5" host01
```

Things become even easier when using default sizes, which is the full-screen. The following opens a VNC session, which is the default for the "-t"- sessionType-option.

```
ctys -b on -a cREaTe='l:CONSOLE'  -g ":5" root@host01
```

or

```
ctys -b on -a CrReAtE='l:CONSOLE' -g ":Screen5" root@host01
```

or

```
ctys -b on -a CrReAtE='l:CONSOLE' -g ":Screen5" -l root host01
```

REMARK: Keywords are case-insensitive and handled internally with uppercase for keywords only.

The previous examples open a session window of a vncviewer for a VNC session and set the title to "CONSOLE". The login is performed as the user "root@host01" of course.

Another build-in feature of VNC-sessions is the stateless operation, and the automatic session-takeover by another login. Thus the shift of a Client-Window to another position on screen or to another machine is performed on-the-fly when the new one is opened. No additional user interaction for takeover is required.

Following opens the same window on "Screen3" and closes old one on "Screen5" with new sizes.

```
ctys -b on -a create=connect,id:1 -g "300x300+500+600:3" \
root@host01
```

Conditional connect or - if not present - creation is supported too by the "REUSE" flag of the CREATE-mode.

Following creates a new window on "Screen4"(assuming this is a new session).

```
ctys -b on -a create=l:TST1,REUSE -g "300x300+500+600:4" host01
```

Following opens the same session in another window on "Screen5" and closes old one on "Screen4" with new sizes.

```
ctys -b on -a create=l:TST1,REUSE -g "300x300+500+600:5" host01
```

That's it related to desktop layout.

5.4 Bulk Access

This features enable the access of multiple targets with one call. The build features are particularly helpful for installation or update of this tool, start standard sessions - e.g. CONSOLE - on multiple hosts for the current(which is DEFAULT) or a given user. When "-t" option is not supplied the default "VNC" is used.

```
ctys -b on -a create=l:CONSOLE -l user01 host01 host02 host03
```

Another approach is to start a defined number of sessions on each machine:

REMARK: The number is currently limited to maximum=20. Yes, ... this piece of software is buggy - as any other.

So, an erroneous count - e.g. of 1000 - could have effects on the target host which you might not agree, particularly when using it on a critical system. Be aware of this when changing the limit.

I accidentally started some hundreds during tests, which unfortunately happens so fast that nothing else than a "hard-reboot" had agreed to be performed within this century...

```
ctys -b on -a create=10,l:CONSOLE -L SERVERONLY \
host01 host02 host03
```

Which makes 30 sessions for current user, but due to "-L" option no clients are started. Just the VNCserver is executed. The label will be indexed in this case by an incremental postfix unique within current session.

```
ctys -b on -a create=connect,id:1,id:2,l:CONSOLE,id:9 \
host01 host02
```

This connects to the sessions on each of given host.

Each host could be set individually with any option available, particularly individual "geometry" parameters are supported.

For any VM based session several addressing schemas are supported, so e.g. the following call searches a given base directory for vmx-file for the given UUID and when matching it the VM will be started:

```
ctys -t vmw \
-a create="uuid:01123...123",base:vmware/dir2,RECONNECT \
-D admin \
-g 800x500+100+400:5 \
-L CONNECTIONFORWARDING \
-W \
host01
```

That's what happens behind the scenes:

- Due to RECONNECT any locally running client will be terminated, on the server(host01) and on the callers machine.
- On the caller machine, due to the "-L" option causing the client to be executed on the local machine and "digging an encrypted tunnel" to the server.

REMARK: Currently requires VMware-Server locally and remote, will be checked during init, and rejected if not.

- The window of the client will be started on the desktop named by the user as "admin" due to "-D" option. Alternatively the numerical ID could be used.
- The size and position of the window will be set by "-g" option as defined by geometryExtended, which is X11 geometry semantics(for now only '+' are allowed). In addition the screen number of "/etc/X11/xorg.conf" will be used for the offset. Following an display of the clients window with size "800x500" on screen 5 with the offset of "+100+400".

  The server resolution is set by default to the viewer size.

REMARK: Requires the deactivation of Edit->Preferences AutofitWindow and AutofitGuest, otherwise the position is only set.

Almost any supported option could be set for specific hosts only, superposing the current state:

```
ctys -t vmw \
-a create="uuid:01123...123",base:vmware/dir2,RECONNECT \
-D admin \
-g 800x500+100+400:5 \
-L CONNECTIONFORWARDING \
-W \
-- \
host01 \
host02'(-t vnc -a create=connect,i:2 -g :4)' \
host03'(-t vnc -a create=connect,l:CONSOLE \
-g :3 -r 1600x1280)' \
host04'(-t vnc -a create=l:TST,REUSE -L DF)'\
host05'(-t vnc -a create=l:TST -L SO -r 1280x1024 -D 1)'
```

- host01 Created as before.
- host02 Connects a local vncviewer to VNC session with ":2", and opens a window on desktop "admin" screen #4.
- host03 Connects to VNC server with LABEL="CONSOLE" and opens on screen #3. The server resolution is set to "1600x1280", whereas the viewer size remains as "800x500", thus scrollbars appear.
- host04 Creates if not present, a 3 new sessions, naming them "TST001, TST002, TST003" and running the vncviewer with "DF=DISPLAYFORWARDING". If e.g. the session ":2" is already present it will be reused, else created.

  The server resolution is for newly created sessions still set to "1600x1280", whereas the viewer size remains as "800x500", thus scrollbars appear. This is due to chained superposing without reset.
- host05 Similar to host04, but now no client is started (SO=SERVERONLY) and already present servers lead to an abrupt cancel of execution. The already created sessions still continue to be executed, no automatic cancel for partial jobs is applied.

  The window will be displayed on desktop "#1", which might be different or not from the desktop as labeled "admin" by the

user.

For additional examples refer to the chapter "EXAMPLES-BASE" or use the online help for displaying the EXAMPLES only:

```
ctys -H "EXAMPLES
```

For additional description refer to the options itself.

5.5 Encryption and Tunneling with SSH

Due to the the "forest" of interconnection types of several tools and their requirements, the decision for this tool was made to support encryption by OpenSSH only. The commercial version might work appropriately too, even tough it is not tested until this version.

REMARK: First of all, this tool is intended for handling user interfaces and some automated sessions with less or no priority on the ability of handling bulk data. The focus is more on the handling of display data, which is partly optimized for line bandwidth not required to speak about nowadays. The connection establishment, as long as it offers an adequate response time for a user interface, is not that critical too. Even the bulk configuration of sessions with huge numbers of logins requiring a recognisable time, may seem to a user as performance boost, who is accustomed to do that manually.

So, the usage of OpenSSH sessions with dynamic allocation would be for the target use-cases perfectly all right.

The first pro-argument was the broad support of platforms and tool-complements for OpenSSH, particularly the easy utilization and seamless integration into KerberosV. It works perfectly with combinations of MIT-Kerberos and Heimdal implementation. Additionally the seamless integration and interworking with GSSAPI based OpenLDAP access and the LDAP based automount for SSO completed the decision.

The integration with the X11-based user interfaces supported by the embedded port forwarding feature of OpenSSH was the second main argument to use it.

The next decision made was not to support SSH specific parameters of some tools too. This decision is again made in order of extracting a reliable common standard interface and might be extended in future releases, when appropriate.

Thus currently the following two constellations of encrypted communication channels with the related port-forwarding are supported.

5.5.1 DISPLAYFORWARDING

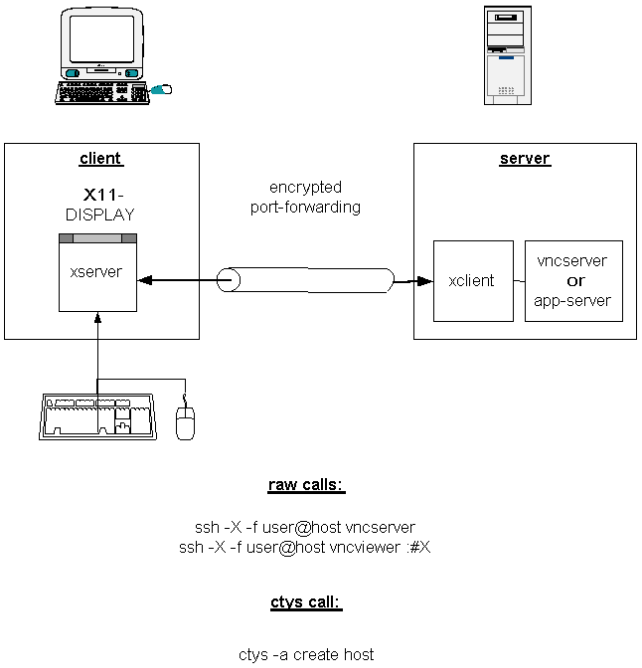

Figure 5.11: DISPLAYFORWARDING

DISPLAYFORWARDING

The whole set of applications processes including the virtual peer-Xserver of the application will be performed (virtually) on the server, whereas the client is in this case conceptually a ThinClient.

Concerning the application of SSH, the connection will be established as a channel of an active user session and will be closed when the user closes his session.

No additional precautions for the termination of the ssh-tunnel is required.

5.5.2 CONNECTIONFORWARDING

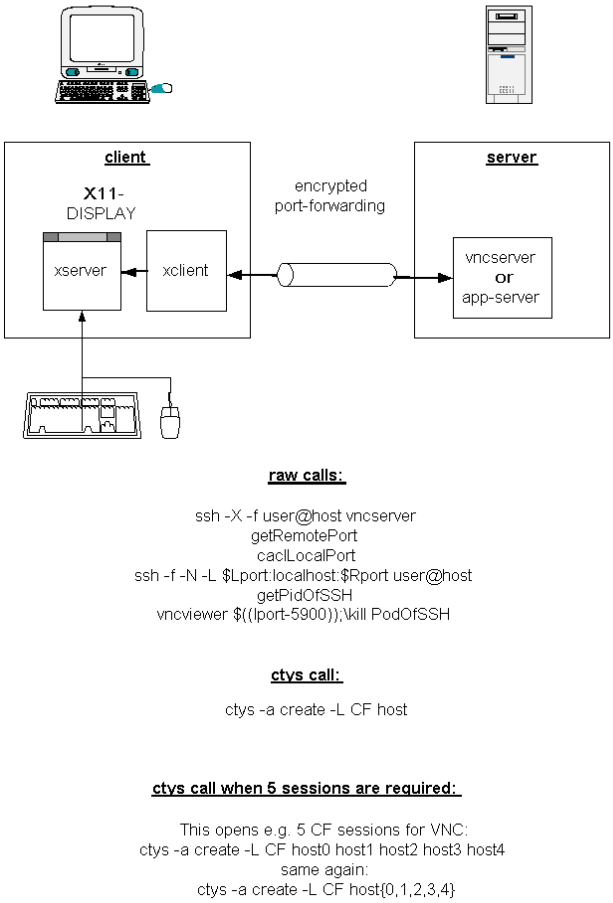

Figure 5.12: CONNECTIONFORWARDING

CONNECTIONFORWARDING
: The advantage - if required so - is in this case, that the client process is executed on the client machine and the communications by the applications protocol only will be forwarded to the server-process/machine. In case of efficient protocols this is clearly an advantage for restricted-bandwidth communications channels.

REMARK: Why is it called ConnectionForwarding instead of the common term Port Forwarding?

Yes, this is just an abstract, not related to whether a Forwarding("-L CONNECTIONFORWARDING"), "Backwarding"("-L DisplayFrowarding"), a dynamic or static approach is used, it just means:

1. The application executes a client locally.
2. This client establishes a connection based on applications protocol to it's remote server process.
3. The connection utilizes an ssh-tunnel.

Mainly two drawbacks result of this:

1. The usage of Connection Forwarding requires the establishment of an ssh-channel explicitly. This results in some port calculations in a multi-client-to-multi-server environment. The requirement is caused by the local-only uniqueness of the port assignments on the involved nodes for the majority (if not all) of involved applications/tools.

   For example VNC enumerates it's DISPLAY-Numbers for each node incremental beginning with one. This numbers are used for the servers listening communications ports as an offset e.g. to 5900.

   This port and/or DISPLAY number could be set by the user more or less arbitrarily too, but as mentioned, this requires the management of numbers. In a multi-user and multi-node environment the assignment of unambiguous numbers gets quickly cumbersome.

   So, for this tool a combined approach is choosed. The remote numbers will be managed by the hosting nodes locally, the local port/display numbers for the client will be calculated as unique peer-access-points.

2. The second drawback is the required explicit cancellation of the port-forwarding tunnel. This could again be defined in several ways. For now and this tool a complete dynamic allocation of session-attached allocation of ssh-tunnels is defined. Thus these will be terminated with the session consequently.

   The choosed solution gets clearer when examining a non-shared VNC session, which is handed-over by opening a session to an

existing one from another node. In this case, which is perfectly all right and for a roaming user not unusual, the trigger for cancellation of the ssh-channel has to be given by the hand-over of the vncviewer. This is coupled with the release of the port/display number, which now might be re-used on the remote node for another session, which could be owned by another user from another node. So, the local channel might not be reuseable at all.

So, the cancellation of the channel with the related session allows last but not least for an easier algorithm of local port/display assignment too.

5.5.3 Execution-Locations

The following options are related to control of clients execution, execution-location, and interconnection.

-L "CONNECTIONFORWARDING" Executing a local client and forwarding its port from local access to the remote by SSH(ssh -f -L).

-L "DISPLAYFORWARDING" Executing client and server on remote machine and just forwarding the display to local XServer by SSH(ssh -f -X).

-L "DisplayRedirection" This case is similar to Display Forwarding, but redirects the display to another XServer running locally. Remote redirections are withdrawn. The primary intention is to concentrate consoles of multiple sessions started independently within the same PM on specified virtual desktops like VNC.

REMARK: This feature is currently under developement and might not yet be available.

-L "SERVERONLY" Starts the server only, no client is started. A client could be attached later by '-a connect=...' option. This could be performed either by Display Forwarding or by Connection Forwarding.

-L "LOCALONLY" Starts the server and the client on local callers machine. So no remote connections and thus no SSH is applied.

Chapter 6

Advanced Features

In addition to the standard CLI for addressing explicitly only one specific task by all it's parameters some features are provided, which ease the daily usage. These are mainly

Group Objects
: for handling sets of targets by an alias comprising the whole set

Customized interfaces by MACROs and TABLEs

Parallel and Background Processing for Bulk Targets

6.1 Bulk Access

Additional to supplying multiple targets just called one by one, ctys supports the concept of groups . A group could be simply defined by creation of an ordinary file containing an arbitrary number of host names or nested include assignments.

The literal name of the group file could be used as a fully functional replacement of any <execution-target>. Multiple group names are supported as well as intermixed group names with host names.

6.2 CLI-MACROS

The MACRO feature supports the usage of a predefined string alias as a literal replacement within any position of the CLI call.

Thus a macro can contain any part of a call except the command itself. The whole set of required options including the execution

target or only a subset of options could be stored within a macro in a defined file. MACROs could be nested and chained as required.

6.3 Generic Custom Tables

Several actions, particularly the GENERIC class of calls LIST, ENUMERATE, SHOW, INFO support data to be displayed in multiple specific views. The same applies for some support tools, particularly "ctys-vhost". The views may vary form task to task and should emphasize different topics.

Therefore the output could be adapted by the user with generic tables, which support a simple syntax with required minor knowledge only. These custom calls, which are based on a suboption for the specific action, could be stored as a MACRO and reused later by it's shortcut.

The recursive MACRO resolution supports for modularized table definitions which could be reused within the same ACTION, but due to canonical standard parts of some ACTIONS as LIST and ENUMERATE, also partly within multiple ACTIONS.

6.4 Parallel and Background Operations

The implementation of ctys supports for several measures in order to enhance the overall performance and reduce the individual response time.

The data to be handled by ctys is actually of two different characteristics, pure dynamic, and static. These aspects are coverd by the LIST and ENUMERATE action. Where the LIST action displays the primarily dynamic data, almos in realtime or better in neartime. The static data is displayed by the ENUMERATE action, which handles data preconfigured by the user, representing an almost static VM.

Despite the nature of the data, the collection of the instances could be designed with various concepts. The data collector could operate sequential or parallel, can do this in a synchronous or asynchronous mode, occupy the callers teminal in forground mode or release it when going to background mode. Additionally the data could be cached for combined operations on multiple targets, or just for a

later reusage.

The UnifiedSessionsManager supports all of this. The two relevant flags are "-b" and "-C" for the ctys call, which control the combination of these operational modes.

The user defined setting of these flags might not be required often, because for each action an almost for any case appropriate default is pre-set.

Some demonstrations of performance impacts with wrong settings are given in the Section 27 'Performance Measures' on page 555 .

A specific ordering effect for displayed data occurs for collecting actions with prefix output to be displayed, when no caching is active. Examples for this are LIST , ENUMERATE , SHOW , and INFO , when a table-header has to be displayed first. Therefore the following has to be considered:

- The basic operation, when no file-caching is active, is to display results immediately within the callers terminal. This could be intermixed in case of multiple jobs executed in parallel . Therefore the output is performed in units of lines, where each line is a seperate record of the output table.

- Due to the immediate display, a required preamble, e.g. a table-header, has to be displayed BEFORE the start of the collector-jobs when no file-caching is active. This is proceeded within the PROLOGUE section of jobs.

  As a result from this behaviour any output of the collector-jobs is displayed AFTER the preamble, e.g. the table-header. Any password request dialogue is displayed also AFTER the table header. So the display of requested and non-requested output from the several collector-jobs is intermixed after the display of the preamble.

  This inherent behaviour can only be changed with usage of a file-cache.

  Anyhow, the file-cache has the prevailing condition, that the data is started to be displayed, and will be displayed at once,

AFTER all collecting-jobs has been finished. This is on the other hand the inherent behaviour of a cached output.

- When file-caching is active("-C raw"), the display of data is performed AFTER the completion of all collecting-jobs, therefore the preamble, e.g. the table-header is printed within the EPILOGUE section of the job.

  The main advantage of this is the collected output of any dialogue and error messages, including any password requests of remote clients, BEFORE the requested output, including a preamble like a table-header.

Chapter 7

HOSTs - The Native Access to OS

Host sessions are gereal purpose sessions connecting a user with any active instance of an OS. Therefore no seperate VM needs to be started. A HOSTs session is an ordinary login with some additional processes started as required. The available standard HOSTs sessions are CLI, X11, and VNC.

All of these are supported to be accessed via a SSH encryption only. Due to standardisation no specific parameters for variation of the encryption are supported, the callee is executed by ssh encapsulation only.

7.1 Command Line Access - CLI

The CLI plugin supports a command line interface, which is executed native within the calling terminal, no new window is required, nor supported. Additional command execution is supported by specific suboptions. Thus this plugin supports for interactive sessions synchronous execution only, even though it is principally possible to use it asynchronously.

The execution of remote commands by CLI could be performed more senseful, when the intermixing of output is not a drawback. The particular advantage is the usability from ASC-II terminals only, which do not support a graphical user interface. Therefore no graphic option is applicable for CLI itself, thus the type of forwarding is not relevant here for now.

The only supported connection type is an encrypted ssh connection, thus the CLI module is more ore less an ssh connection with additional ctys-addressing capabilities.

7.2 Start a GUI application - X11

The X11 plugin supports the start of GUI based terminaly or gerenally GUI based applications. Therefore any command could be provided and will be executed on the provided execution target.

Due to the stateful operations of X11 client and server architecture "DISPLAYFORWARDING" is the only supported operational mode.

The asynchronous execution mode is the applied standard.

The only supported connection type is an encrypted SSH channel with local access of X11 protocol only. Thus with the usage of ctys any X11 connection becomes a secure remote connection with encryption provided by ssh.

7.3 Open a complete remote Desktop - VNC

The VNC package supports for complete remote desktops currently based on RealVNC(TM) and the derived project TightVNC.

The software packages RealVNC(TM) and TightVNC are based on a open protocol definition and the implementing tools. This is important to recognise, due to the wide support of a so called VNCserver interface. The frame buffer protocol used for the efficient transformation of an "almost relatime" desktop view, is supported by a wide range of native remote desktop interfaces as well as by a huge number of VMs for various kinds of CONSOLE types. This interface is particularly essential as client interfaces for the VM plugins XEN and QEMU.

Due to the splitted client and server components this plugin supports "DISPLAYFORWARDING" and "CONNECTIONFORWARDING". It could be used in this manner in combination with any supporting VM. For example the frequently required remote ports which occurs as the unavoidable very first question when trying to apply VNC,

could be statically cuytomized or dynamically allocated.

In order to provide a common and generic interface the encryption and DISPLAY handling is designed as a seperate facility. The handling of communications is generally supported by the seperate package OpenSSH[119] only, thus no specific wrapper-interface is supported. Despite some differences [26, sshDefGuide] SSH Tectia(TM)[121] may work too, but is not verified.

Chapter 8

PMs and VMs - The Stacked-Sessions

The previous aspect of creating generic remote sessions to local and remote hosts is closely related to remote start of virtual machines. In nowadays the distinction between physical and virtual machines is getting lost. OSs like OpenBSD, NetBSD, or Linux could be even installed in a VM running in a UNIX/Linux environment, customized and tailored as required, burned on a DVD or stored to Flash-Memory, and boot seemlessly with only minor previous changes, running on almost any hardware.

The management of logical sessions to running OSs, either on a "Physical Machine" - PM or a "Virtual Machine" - VM is the main task of the UnifiedSessionsManager. The differentiation from all (known) existing solutions is the extended and formalized usage of stacked VMs and PMs, where VMs are nested to logical stacks. Thus the offered logical and integrated addressing of stacked VMs is the core advance uniquely provided by UnifiedSessionsManager for now. It could be said, that the overall feature scope, particularly the integration as implemented and availabale in present version seems to be the only and one available solution on the market since first publishing at 10/2007 until now(05/2008).

Even though the construction of a logical stack of "physically" nested processes of VMs seems to imply a deep nested-subprocess-structure with proprietary internal task-schedulers, this actually is not the case. A modern OS should be, and is, able to dispatch it's processes and threads to any of the available CPU cores by various criterias. Thus a nested VM stack is actually performed as a flat set of processes dispatched to available CPU cores, but addresses in

smart manner as a logical tree with groups of branches and leafs. This fact becomes of real relevance, once the number of CPU cores is raised far above of 4 as available for now. When the number of CPU cores raises, the software design could be extended by the paradigm of "Virtual Components" - "v-components", which are complete appliances used as encapsulated software modules with a communications interface only. Depending on the type of the utilized hypervisor, these could be fully standalone VMs including the whole GuestOS, or just a reduced GuestOS-stub for host-kernel based VMs.

Another approach which is partly available, is the setup of physical distributed CPUs to a logical "single multi-core CPU" as a Virtual-Server. This approach is not extended within the scope of this document for now.

Anyhow, current available CPUs with up to Quad-Core versions already provide means which are definetly capable for several types of production environments which fit for various specific applications.

Almost unlimited sizes of environments are applicable when a flat or just two-layered(PM/Xen-Dom0+DomU) environment without(or less) emulation is utilized. This is the common case for nowadays and fit's for pure partitioning of machines with some failover resizing measures perfectly. It is a quite good solution for simple encapsulation too. Within development departments this is the perfect solution for setting up advanced and huge sized development and test environments.

The actual stacking with emulated CPUs within another hypervisor, e.g. Xen, VMware, or KQEMU/QEMU might be approriate on PC based machines for small to medium sized systems only, which for sure will be extended soon. Typical applications in this field are Cross-Compiling and Cross-Development in a "native" environment. Various CPU emulations are available by QEMU.

The following stack models are supported beneath a PM-only model by the current version of UnifiedSessionsManager.

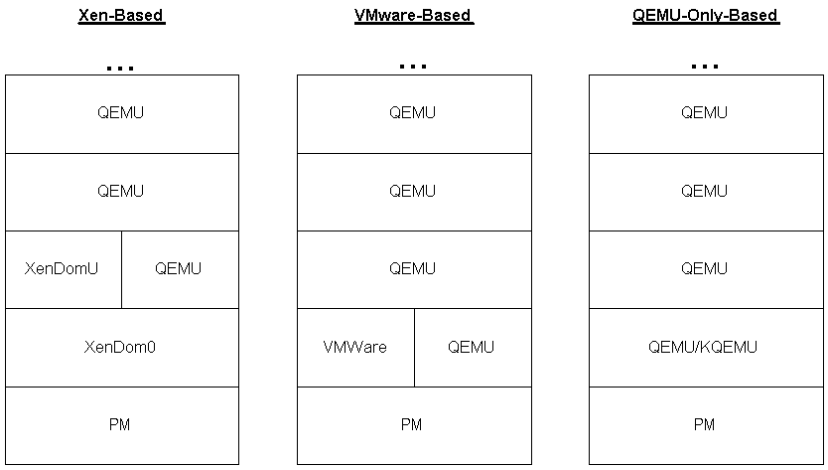

Figure 8.1: Supported Stackmodels

Within this model each depicted upper layer element could be present in multiple instances and can contain multiple instances of it's upper-peer itself. Whereas it is contained in one lower-peer only, of course. Thus this sets up a resulting tree structure.

Some hypervisors accessing Ring-0 components could be combined under specific conditions, but are just draftly tested here. The basic idea is to combine "uncritical" components for daily-production usage which could be maintained easily. Therefore the QEMU emulator is combined onyl with a "kernel-touching" component, for the first release with one of:KQEMU/QEMU, Xen, VMware(S/P/W)". The additional constraint for the actual available components is the only applicability of a hypervisor as the bottom element, whereas emulators could be stacked above. Performance could be - not neccessarily has to (!) - become a thread.

The actually implemented and tested models are "Xen-Based" and "VMware-Based", which include "QEMU-Only-Based", when KQEMU is not applied. For the required proof-of-concept only, the performance drawback of the absence of KQEMU on i386 was not an issue. The actual production environment requires for several reasons "VMware-Based" and "Xen-Based" anyhow.

8.1 Session-Types

When defining a system with various sessions, which comprise pure HOSTs sessions for console access, PMs sesssions, which manage basic entities containing others like HOSTs or VMs, or VMs containing a full-scope virtualized entitiy, the distinction of several session types is required for several reasons.

Obviously a modular and extendable software design could be grouped into components associated to several sets of functionality constituting a session type. Additionally some session types like Xen or QEMU do not have their own frontend for console and/or GUI access, thus for the actual design any of the so called HOSTs plugins could be combined with any other plugin of types PMs or VMs for user access.

Another aspect is the overall design of a distributed access and resulting client-server architecture with probable application of virtual-circuits crossing multiple intermediate SSH-hops. This scenario is particularly helpful when "piercing" a firewall via a gateway within one of it's DMZ.

The general design of the UnifiedSessionsManager is a single-code implementation as a generic application which serves in a holomorphic manner for any required communications entity within the peer-to-peer circuit insances. Therefore the called client is exactly the same code as the serving server instance. The whole process structure is implemented as an on-demand-executable, thus no daemons are required. The UnifiedSessionsManager serves as a distributed dispatcher and starter with basic management capabilities for cancel, list and info. Some of the plugins are capable of remote execution of any own library call or any external call when appropriate permissions are present.

Therefore the design requires some clear distinction between session types and their capabilities when utilized within the various instances of a communications chain. For example a Xen session type behaves on the caller site, where probably nothing specific is executed, completely different than on the server site, where for example a DomU will be started and a console - of another session type - is opened and just redirected by display-forwarding.

The interface for the definition of a session type is called "plugins"

and alows for (almost) easy implementation of any additional custom type. The integration is implemented by an automatic runtime detection of present plugins and a self-initialisation of the components. Generic actions like LIST and ENUMERATE are caller by wildcard mechanisms based on a few name conventions for interface functions.

Following session types, a.k.a. plugins are available for current version.

PM/VM	Description
CLI	Command Line Interface.
X11	X11 caller.
VNC	Native or virtual running OS accessed with RealVNC or TightVNC.
QEMU	With VNC, X11, or CLI modules as remote console. Additionally the SDL console is supported.
VMW	VMware workstation, server, and player. With it's own proprietary console or VNC client.
XEN	With VNC, X11, or CLI modules as remote console.
PM	Physical machine access, supports Wake-On-LAN.
TUNNEL	Internal subsystem supports implicit CREATE and LIST of OpenSSH tunnels for distributed Client/Server-Split by CONNECTIONFORWARDING .

Table 8.1: List of Standard Plugins

For additional plugins refer to specific help of generic loaded modules.

The handling of the virtual machines is controlled mainly by the parameters "-t", "-T" and "-a". Where "-t" defines the type of virtual and/or physical machine to be addressed, "-T" the set of plugins to be loaded, which could be required additionally as subcalls or focus of operations for generic collectors. For example LIST with "-T all" lists all actual running instances, whereas with "-t vnc" or "-T vnc" onyl VNC session types are listed.

Due to an implemented dynamic load environment for bash plugins, only required plugins are loaded by pre-defined static link-lists or dynamic by on-demand load. This is controlled by the "-t/-T" options too. Refer to development documents for additional information.

-t one type of session to be actually performed

-T list of session types to be preloaded, e.g. as supporting subfeatures, particularly as a specific type of console application.

The "-a" parameter defines the action to be performed within the choosen session type.

-a mode of action or access
For additional information refer to following options descriptions.

The handling of the sessions is - as mentioned above - splitted into a client and into a server part. Therefore ctys will be executed locally and remotely, with specific context provided by "-E". The executable itself has to be literally the same due to compatibility issues, this will be assured by exchange version information. Some minor version-jitter may be accepted, but is an internal feature only.

Therefore ctys has to be installed in compatible version on all participating peers. When using a network account with a central HOME directory auto-mounted on each target, the tool does not need to be copied, same for eventually accessed VM configuration within the own directory tree.

The various session types, including all categories, have the same basic interface. The interfaces vary by their suboptions, which is required due to specific differences. Thus all support the actions CREATE and CANCEL as their own exclusive top-level methods. The exception here is the CANCEL method, which is not supported within current version for CLI and X11 types as pure "console".

Additionally to actions on specific session types generic actions are available, which have sub-dispatchers for calling sets of session types for display purposes, and generic methods for retrival of information for specific target, which includes basic information about the type of sessions plugin itself.

The generic sessions LIST and ENUMERATE work as collectors for a given set of sessions, which could be displayed intermixed when requested by the user. LIST shows the current actual runtime instances, whereas ENUMERATE collects information for stored sessions from their configuration files.

The handling of usage of VNC as console client only for XEN, QEMU, and VMware is controlled by the offered usage-variant only, not within the VNC plugin.

The actions INFO and SHOW display common data for the addressed target itself, thus the data shown for session types - a.k.a. plugins - is the information of the operability of the plugin itself, not the managed sessions.

INFO displays static data, such as version information and installed features. This includes the runtime basis, which is the base OS and the HW. Therefore for example the virtualization capabilities of the CPU (vmx, svm, and pae) are listed in companion with the RAM, and some selected management applications like lm_sensors, hddtemp, and gkrellm.

SHOW displays the dynamic data, which e.g. includes a one-shot display of top output. The health and by default present alarms as given by sensors are displayed too.

8.2 VM-Stacks - Nested VMs

8.2.1 Stacked-Operations

The operational environment of ctys is mainly focused on usage of stacked VMs, which are actually nested VMs. Therefore the provided plugins support silently iterated operations on running entities for actions where appropriate.

This means for example, that if the PM, which is the host anything is physical located on and therefore "could be said depends on", will be CANCELed, the stack of VMs and HOSTs sessions will be handeled by forward - here upward - propagation of the CANCEL call. This will be handeled properly for any intermix, and of course has to be supported for custom made plugins too. But a single and simple call interface is all to be used.

In case of CREATE basically the same propagation mechanism is provided, whereas logical-forward means virtually downward for CREATE.

The stack-propagation strategy could be controlled by the common sub-options FORCE and STACK. For details refer to the following chapters and the related options.

It should be mentioned here, that some specifics occur for generic actions in combination with stacked operations.

The ENUMERATE method, which is a static collector, does not support stacked operations for now. This is due to the requirement of a running instance to be ENUMERATEd of it's contained sub-instances. Thus the ENUMERATE method has to activate the whole upper stack - permutated of course - when would be applied with automatic stack resolution. This seems not to be a practical applicable behaviour.

Whereas the LIST method perfectly fits to a stacked operation as real gain of usability and transparency for nested VM stacks, which are "perfectly" encapsulated by definition.

Thus it could be said as a rule of thumb, that static methods are not applicable to implicit stack resolution for real systems due to the resulting bulk-activation. Whereas the dynamic actions support essential benefit to the user on already running instances.

8.2.2 Specification of VM Stacks

One very basic idea behind the usage of virtual machines - VMs and physical machines - PMs in a unified environment is the stacking of multiple instances. "Stacking" in this context is more a nested execution, though the each layer is contained and hidden by it's downstream VM as it's execution environment.

Therefore two views are defined in order to depict the layers and define their dependency. The functional dependency is crucial for recovery functionality such as startup and shutdown.

The primary model visualization is similar to the B-ISDN model description by usage of panes. In the context of VM stacks the definition is given as:

- vertical front view:
  depicts PMs and VMs only
- vertical side view:
  depicts the stack of OSs
- top pane:
  depicts for each layer it's contents

The following figure shows the pane view as a "3-dimensional" blueprint.

sl-0: The front-view lists the stack as nested containment hierarchy. Where on a PC-Base - the PM-plugin - the OS Linux is operated, as depicted on the side-view. In this case the Dom0 of Xen.

sl-1: The next layer is the an arbitrary DomU operating the OS Linux as depicted on the side-view.

sl-2: The following layer is an "in VM" operated VM, which is a User-Space process emulating an ARM-CPU.

The top-pane, which is the only visible, here shows the running entities os the topmost VM and it's operated OS. In this case it is an arbitrary application based on test packages of QEMU-0.9.1 - "arm-test". I is executed on two layers of CentOS-5.0 in an Xen-3.0.3 environment.

Just for completeness, the compilation of the QEMU version, due it's gcc-3x requirement, was performed within a SuSE-9.3 linux installed in a VMware-WS-6 version as a 32bit machine on a 64bit CentOS running on a dual Opteron Server. The "make install" does not require a gcc-3x, and could be performed with the standard gcc-4 installation on CentOS-5.0.
The arm-test version with ARM-Linux works from the box, but requires some network configuration. The main pre-requisite is the configuration of TAP devices for sl-1, which is described in the examples chapter. Additionally the eth0 device within the arm-test layer sl-2 has to be configured by ifconfig only, which is sufficient.

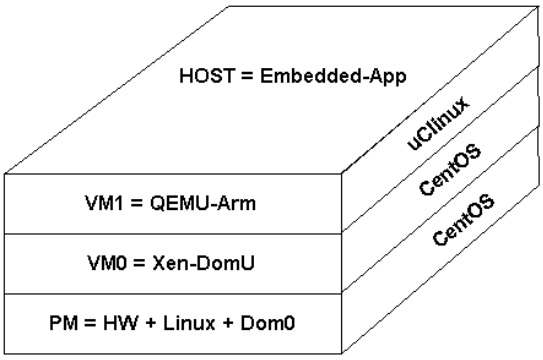

Figure 8.2: Pane-View: QEMU-ARM in Xen-DomU

The following "2-Dimensional" ISO-like stack blueprint shows the dependency in a more close view to peer-to-peer dependencies, required, when implementing protocols for stack-management, which is very similar to implementation of an communications protocol.

This view is for example particularly helpful, when any kind of propagation has to be implemented, as in case of a stack-shutdown.

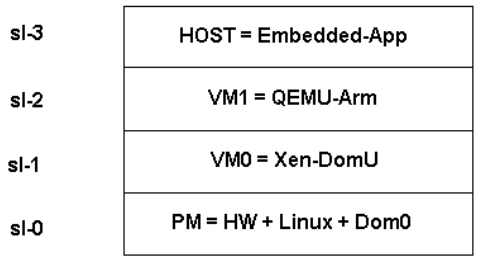

Figure 8.3: Stack-View: QEMU-ARM in Xen-DomU

This is tested with the provided test-cases of QEMU and additional configuration scripts as provided within the templates directory. The cases coldfire, small, linux, and sparc are tested too. In order to interconnect that cases it is required to set install and configure VDE and setup appropriate TAN devices with their interconnecting vde_switch. The IP addresses of the eth0 device within the GuestOSs has to be set appropriately.

For additional information refer to the plugins chapter and the examples list.

The following case depicts the more common case for nowadays, where just one layer of VMs is operated on a host - e.g. Linux. Multiple instances are operated within the only layer primarily for server consolidation an supply of centralized services, e.g. for license sharing.

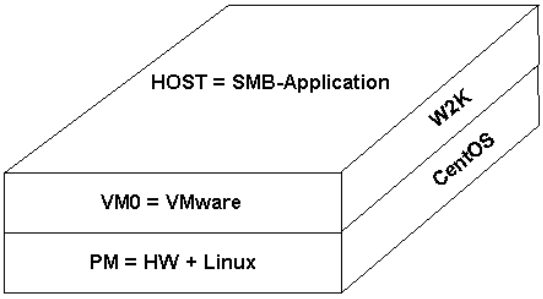

Figure 8.4: Pane-View: W2K in VMware on Linux

The related stack-view shows the communications dependencies for inter-layer management.

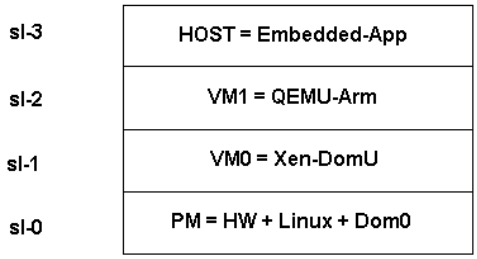

Figure 8.5: Stack-View: W2K in VMware on Linux

It should be recognized, that there are for now 3 models and more or less completely different interfaces for the listed VMs.

Whereas Xen almost has no direct view from the Dom0 to the DomU, just by specific tools, the VMware-VMs and QEMU-VMs could be listed by the ordinary "ps" command.

One particular advantage, which has some performance drawback, is the complete CPU emulation of QEMU. This makes it to the perfect entity for a stacked application. The kqemu module is for mixed CPU types not usable anyhow. This is e.g. one another difference, that QEMU could be used within user space only for a complete emulation, the performance issue might be solved within the next generations of CPUs.

The next difference is the CONSOLE access, which is proprietary with it's own dispatcher for VMware(or a static VNC port for WS6), whereas for XEN seperated VNC ports accessible by independent "binds" exist.

But all of them fit into the model perfectly and are almost unified for their access by "ctys".

The previous figure as shown depicts the basic constellation for an nowadays common configuration, where a single layer of VM on a PM supports a Cross-Platform functionality.

The Nested execution of VMs is officially "almost" not supported, by any VM, just by some, providing their own hypervisor as base only.

The following figure depicts a constellation which successfully executed in lab. Even though the performance was more than limited, the innermost boot finished after real-longer-while. But anyhow, a successful login on the whole stack was performed.

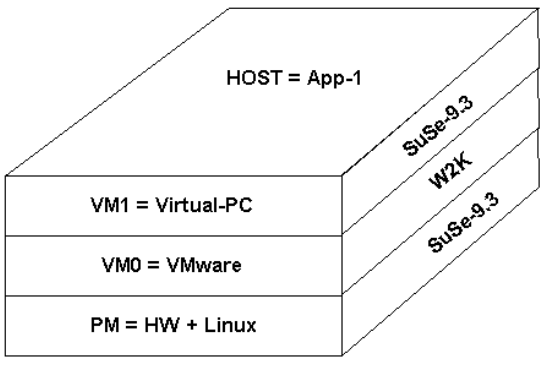

Figure 8.6: Pane-View: Virtual PC with Linux in VMware

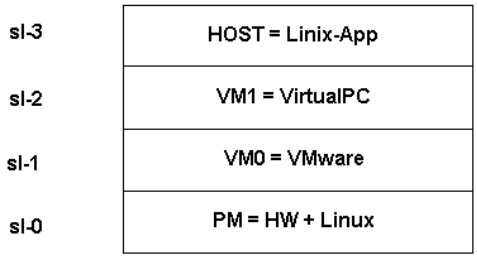

Figure 8.7: Stack-View: Virtual PC with Linux in VMware

Another field of application for nested VMs is the so called Embedded segment where controller based applications dominate. Much of the applications are based on raw, but meanwhile on full-scale but lean Embedded-OSs. Some examples for this are the eCos and the uCLinux project. OpenBSD for example could be used for ROM/DVD based applications perfectly too, as many other variants.

The supported standard integration of QEMU offers ARM, MIPS, Coldfire, PPC, and SPARC CPUs. Additionally some specific Evaluation Boards are modeled. For further information refer to QEMU documentation.

So this might give an idea of what is upcoming within a not too long of period for shure.

Once someone is getting familiar to the idea of stacked VMs as a common SW-component, something similar to a deamon service, the opportunities for resulting SW architecture and design might be almost unlimited.

One trivial aspect is the encapsulation of services by a more than clear message interface, which might be a simple TCP/IP message protocol. This will be optimized for local access by almost any available TCP/IP-stack.

But the first real benefit for modularization now arises from the widely support of physical online-relocation of VMs by almost any current known Vendor and OpenSource-Project.

In difference to an ordinary daemon supporting non-relocatable ser-

vices for example as DNS/Bind, DHCP, LDAP, or a DB-server, a VM offers now a framework, which could be utilized for online relocation of any contained service, which even will not recognize it's relocation.

For a service provider supporting services with SLAs this will be the what is required. Almost ANY proprietary service could be configured redundant and relocatable with an almost 100availability.

The scalability and online reconfiguration capabilities, as well as energy saving seamless activation and deactivation of parts of physical equipment with on demand redistribution of worker instances is just another example. This will be supported for the whole scope of ctys-stack-addressing with partial addressing. Confusing, yes, but this is what ctys silently provides behind the scenes - NOW. So just a simple path address in a very natural syntax for a VM or HOST within a layered stack is all the user actually has to provide.

Therefore the network itself and it's services now seems to become a much closer application "component" for not highly sophisticated user, than before.

8.2.3 Bulk-Core CPUs

One important fact for the design is the expectation of upcoming for CPUs with much more cores than available now, though offering partitioning capabilities even not available on mainframes of nowadays. The time line could be 5 or more years, but the QuadCore CPUs for now already perfectly offer required performance benefits for first stages of design.

The current available test and development environment with some outdated multi-cpu-single-core CPUs match the requirements even when using some PIII-800MHz-Coppermine for basic tasks. E.g. as a "driver encapsulation" for an undocumented, non-disclosed, and outdated PABX-Monitor.

8.2.4 Almost Seamless Addressing

When using the currently available hypervisors and so called HOSTs sessions for interconnecting, almost any used component requires it's

very own style of addressing connections and components.

Additionally the management of the states of an VM and of course a PM requires it's individual set of tools, each of which has it's own philosophy. Some provide very specific and additionaly multiple versions and variants with different command sets and features.

So a common namenbinding is more a daily user benefit than an academic discussion here. Therefore a generic namebinding was defined, which supports all supported components with only and one unified namebinding schema.

For the seamless and integrated management of VMs, PMs, and contained OSs, which are by definition not aware of the virtual environment, and of course not the containing OS for the VMs hypervisor, a multi-level addressing mechanism with a complete set of required toolset is defined.

The current process is ongoing to file the addressing schema as a standard, with an open final state for now of course.

One specific benefit of the defined name-binding is the eas of management and addressing of stack elements. This is valuable for the communications within an operational and enabled system as well as for recovery procedures such as boot and shutdown, a.k.a. Startup and Shutdown.

8.3 Stacked Networking

The required approach for networking within nested VMs has to pass the packets logically "downward" to the "external exchange point" for distribution of the packets across the physical container of the VM-Stack. For a communications peer contained within the current stack, the virtual circuit will be terminated without leaving teh stack of course. In order to establish a TCP/IP connection each VM sets up a virtual NIC and a virtual interconnection to it's containing peer VM. Additionally the communications facility has to provide multiple users where each user could have multiple VM interconnections.

This requirement seems to be provided most appropriate by the utilization of virtual bridges in combination with TAP devices. This

approach is very close to the networking structure of Xen *Xen-Wiki - Xen-Networking*[114, xenWiki].

Therefore for each VM beginning with a PM a communications facility is recursively setup as depicted in Figure: 8.8.

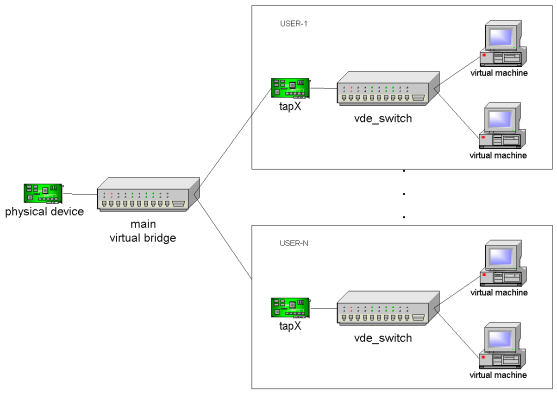

Figure 8.8: Virtual interconnection structure

The implementation of this structure is described in Section 30.1 'TAP/TUN by VDE' on page 579 and is automated by the support-tool Section 18.9 'ctys-setupVDE' on page 356 . This tool supports particularly remote operations, thus the complete required network environment could be configured and removed on demand from a remote station - for multiple physical machines and virtual stack-entities - by just one call.

Once the required access permissions and the presence of the called system tools are in place, the usage of this utility reduces the complete setup and final remove of the additional network environment to a simple call and some seconds of processing.

This structure and the supplied utilities could be used and are prooven to work as a building block to be combined for interconnection of all members of a stack in a recursive manner as depicted in Figure: 8.1.

Later developments might eventually extend the capabilites of this first draft approach.

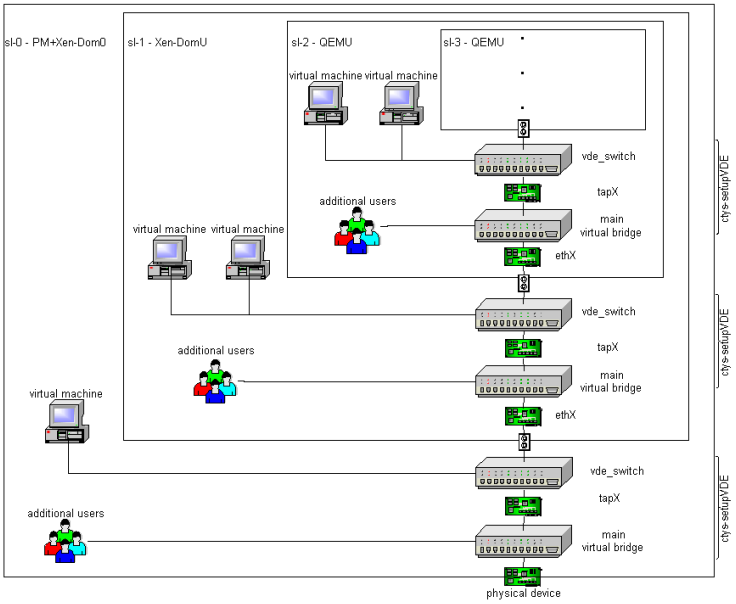

Figure 8.9: Nested Protocol Stacks

8.4 Stacked Functional Interworking

8.4.1 Stack-Address Evaluation

The basic schema of an action to be performed within an stacked entity is given by the common concept of CLI partitioning into the action requested, and clearly sepereated from this the execution target, where the requested action has to be performed.

Even though the following description is very close to a design specification, it is essential to understand the implicit boot procedures of missing pre-requisites, and the resulting dispatch and parallel/background operational modes. Thus the following part is presented here. Additional description with a call example and resulting process structure is presented at Section 13.3.4 'Stacks as Vertical-Subgroups' on page 169 .

The execution flow, as depicted in Figure 10.2 is handeled by the so called generic main dispatcher. This is the part which is called in a chained manner on each actual executing machine and acts as a hierarchical controller structure. The task data resulting of the

requested job and it's expansion is depicted in Figure 10.3.

The main task dispatcher handles the execution target with a generic approach, where it is aware of the address syntax and the optional present context options. The hierarchy and therefore the startup and shutdown dependecy whithin the stack is also known to the generic task dispatcher, but as a common pattern only. Thus the execution base could easily be splitted into optionally pre-required subjobs to be started silenty before executing the actual requested action. Due to the dependency of the order of availability, these pre-startups are executed within a thread in a hard-coded only, whereas several stack-startup-threads could be performed in parallel.

Therefore some basic knowledge of required functionality from the involved plugins - such as required WoL for an initial boot of a PM - is also available to the generic dispatcher.

Anyhow, the actual tasks are performed by the involved plugins, the dispatcher just has the knowledge about whom to call, and how to do that. The specific plugins required on the various levels have to be provided by the context options on each level as the "-t" option.

When the pre-required intermediate stack-elements are already present and operable, the stack-level is simply accepted as ENABLED and ignored for the further processing.

A required startup is monitored by a combination of timeout intervals with a counter for poll-trials whether the required instance is already available. This is due to the lack of a common call-back interface for the supervisor in order of advising it to trigger the start-up of a nested inner instance within the actual guest-OS. Therefore the main dispatcher is the controller instance, which is monitoring and controlling of the startup of the whole requested stack.

8.4.2 Startup

The startup of entries within a stack of VMs will open - once handeled completely - a variety of advantages. One of the most obvious might be the implicit creation of all intermediate sessions below the targeted stack entry, which includes consequently the PM itself if not yet booted.

So the handling of complete stacks by just addressing a higher level

entry could not just open some smart advantages in a variety of scenarios, but also help to save energy by enabling machines on demand only, and therefore opening an easy to use facility for making temporarily shutting them down less complicated.

Support for load distribution and therefore concentrating and tailoring appropriate loads will become by stack-aware addressing an daily task for ordinary users too.

Even though this is seen as an important feature, the automatic and implicit start of stacks is for now shifted to one of the next versions. This is due to several reasons, one is the intermixed call interface of the provided VMs, another is the current priority of stabilizing UnifiedSessionsManager first and bringing it to the audience "now".

Thus this versions supports the creation of VM stacks only manually step by step with iterative calls, what might for now not be a real drawback, due to the average of 3-4 layers.

The support of handling of VMs within a stack is in this version already that much supported, that is could be called smart anyhow.

8.4.3 Shutdown

When handling stacks of nested PMs and VMs , the CANCEL action on a base level will force contained instances to terminate too. Thus a behaviour has to be defined, whether a top-down soft shutdown has to be performed, or a "bottom-up" behaviour of instances, by killing the assigned level without recognition of contained instances. This might be appropriate e.g. in emergency cases.

The handling of embedded instances with their own state might seem to be managed via SUSPEND conveniently, but frequently leads to some RESUME problems due to invalidated network sessions and related transient data.

Another issue is the mandatory remapping of commands when walking on the stack. One obvious example is the REBOOT of the following PM containment stack, where each layer might contain several native applications. Almost the same is true in principle, when dealing with some higher level VMs intermediate layers.

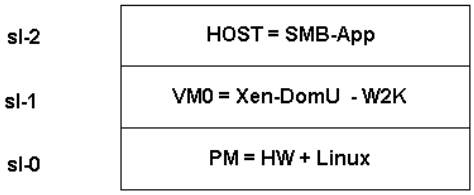

Figure 8.10: Stack-View: W2K as HVM in DomU

When doing a REBOOT on the stack layer "sl-0" a simple forward propagation through the stack in bottom-up direction might not lead to the result expected.

Thus a previous action for disabling the upper layer is required.

The appropriate action still have to be decided between one of the possible "persistently disabling operations".

This could be a stateless or a statefull deactivation.

Even though the upper layers might be in physical and logical state without any reason for termination, and just an unrelated reason on the lower layer might have triggered the required REBOOT, once changed to the offline state, any session to networked peer might lead to protocol timeouts of the involved applications. Thus even a simple SUSPEND, e.g. in case of a Samba, LDAP, Kerberos, and Automount based SSO with an open W2K session within a VM, will not restore completely and requires at least a new login of the user.

As a result some advanced strategies are required in order to end up in a state with previous unrestricted operations mode after accomplishing the REBOOT.

The same will apply to almost any networked protocol with dynamic and non-persistent sessions, where the applications peer is not prepared to continue after a "longer unexpected disconnect". The whole involved communications protocol stack has to support the interrupt of the service. This is particularly true, when a server application connected to multiple clients is located on the local machine. E.g. DHCP, DNS/Bind, or a DB-Server.

The solution choosen for now is somewhat limited but a good base for extension and easy to implement and to apply.

In current implementation the basic philosophy is not to use any persistent system services for runtime-state-management, such as a splitted-reset for upper-layer instances, when rebooting their container. This is targeted to be part of one of the future releases.

Two basic directions are defined:

FORCE Forces the selected instance to be canceled as selected, no previous shutdown of contained instances will be performed. The instance itself may have some shutdown behaviour, e.g. init-scripts, which will be performed unattended.

STACK Uses a recursive approach for shutting down by an top-down behaviour. Therefore the VM-stack will be first walked up form a call entry point of the bottom for the "stack-tree" of upper VMs and embedded GuestOSs. Each instance "on the way" will start an recursive upper call for all it's contained instances and goes into a wait state by a simple sleep call. Once the toplevel instances are reached, the downward roll-out of the actual CANCEL operations starts. This happens for each branch seperately and will be cumulated on the branch-bases to a node-state. Each node itself CANCELs when all upper peers has reached the final state. Where the timeout hits before the upper stack has reached it's final state, the state for the remaining upper peers will be forced by the node or simply ignored.

The propagation of a state is for the basic case of a simple termination of the whole stack quite simple but requires for advanced state-change events some specific treatment. This is e.g. the case for a reset of a VM containing additional VMs itself. Due to simplicity in this version the "inner" VMs do not inherit the initial "outer" state, but are treated by a mapped new state. This mapping is proceeded in accordance to the following table.

Mode	Pre-defined above-modes
PAUSE	PAUSE
SUSPEND	SUSPEND
RESET	POWEROFF
REBOOT	POWEROFF
POWEROFF	POWEROFF
INIT:0	POWEROFF
INIT:1	POWEROFF
INIT:2	ffs.
INIT:3	none
INIT:4	ffs.
INIT:5	none
INIT:6	POWEROFF

Table 8.2: Targets for state propagation of CANCEL action

8.4.4 State-Propagation Basics

State-Propagation

The depicted state-propagation within the previous chapters is beneath the addressing concept the second essential facility required to operate VM-Stacks. This is particularly required due to the inherent encapsulation of the stack-awareness of GuestOSs within each layer of VMs. The willingly designed systems immanent encapsulation is not just for transparent operations but also essential for security reasons. Anyhow, the operations of a VM stack requires for practical reasons some stack awareness, which particulalrly uprises when running services have to be located and relocated to various physical locations. These might particularly depend on some constrins such as physical resources, or just specific configurations for grouped-operations. The CANCEL actions frequently requires the overall controlled shutdown of related groups of systems, which might be assembled to various execution groups within several layer of VM-Stacks.

The stacked operations could be distinguished within a first step into the following basic operations principles,

- Independent Operations
- Hard-Wired Dependent Operations
- Recursively Propagated Operations

The simplest form of stacked operation is the "Independent Operations", which could be operated for independent atoms only. Examples for this are collecting actions like LIST and SHOW, which are defined to be operated on pre-existing entities.

The some more challenging operations mode is the "Hard-Wired Operations" mode, which is implemented as a specific SUBTASK named VMSTACK, refer to Section 13.3.4 ‘Stacks as Vertical-Subgroups’ on page 169 . This mode supports a left-to-right dependant canonical notation, and is due to it’s specific keyword aware of basic inter-stack dependancy. One basic characteristic the VM-STACK is aware of is the consistency and physical nesting of the VM-Stack it contains. Therefore the common <machine-address> and the <execution-target> will be evaluated and checked for consistency due to founting a single VM-Stack. The current version supports a single VM-Stack as a "column" within a tree, wildcards may follow in future versions. The introduction of wildcards may be considered thoroughly due to it’s permutation effects for the following above layers and the uprising redundancy of operated virtual hosts. The VMSTACK supports for intermixed operations with actual hypervisor calls and additional helper sessions like specific HOSTs sessions for openning CONSOLEs. The very specific PM session with support of relayed-WoL with various configurations is compatible with a VMSTACK too, thus the complete activation process including the physical host machine could be performed. A Typical application for the VMSTACK is the creation of a VM-Stack.

The next mode of "Recursively Propagated Operations" is inherently stack-aware and used for automatic propagation of a state within a stack. A typical application of this is the CANCEL action, which naturally has to terminate the nested upper layers of a cancelled VM too. Thus CANCEL propagated the STATE-CHANGE-REQUEST for CANCEL bottom-up, and each branch itself, once the final leaf is reached, implements the STATE-CHANGE by execution of the CANCEL action in top-down direction. Another variant of this is the collection of information for the whole VM-Stack by one initial call. Typical for this might be the recursively propagated LIST action, which lists any session within the whole set of layer of a VM-Stack.

The current first approach is based on handlng of information related to VM-Stacks mainly based on dynamic runtime-data to be

collected and on request short-time cached. Additionally static data is maintained within each VMs configuration in a decentralized manner, but collected by automated tools for usage within multiple centralized cacheDBs, representing various user specific customizable views.

The following table shows the resulting stack-aware methods for the various actions.

ACTION	Method	Propagation-Direction
CREATE	VMSTACK	synchronous bottom-up
CANCEL	RECURSIVE	asynchronous bottom-up + top-down
LIST	RECURSIVE	asynchronous bottom-up
ENUMERATE	tbd.	tbd.
SHOW	tbd.	tbd.
INFO	tbd.	tbd.

Table 8.3: State-Propagation for the first version

The various methods support combinations of various operational application scopes resulting from the provided parameter sets as given by the basic call interface schema

```
ctys -t <action-type> -a <action>=<action-target> <execution-target>
```

The current <action-target> is provided as a single instance only, where some variants with "ALL" exist(see CANCEL). For the <execution-target> single hosts and group instances are supported, additionally context specific options could be provided.

ACTION	Non-Stack	Single-Peer Peer-Lists[1]	Auto-Stack
CREATE	-	x	-
CANCEL	x	x	x
LIST	x	-	x
ENUMERATE	x	-	-
SHOW	x	-	-
INFO	x	-	-

Table 8.4: Application of Propagation Scopes

The various application scopes result tightly from the implied specific addressing modes. Some temporary restrictions apply to the current version due to the choosen simplified implementation, but the basic concepts except the group-object for <machine-address> and some wildcard features are provided by the current design.

The address syntax is provided as a canonical base set, which will be extended to additional views and mappings of address syntaxes.

The current version easily supports several thousand VMs on a bunch of physical machines. The supported performance in flat operations is more than sufficient, the implementation of complex stack-aware features will be improved, but require due to immanent requirements some more processing when designed purely dynamic.

Anyhow, an additional LDAP based version will follow, which might rely on some extended nameservices.

Integration into Nagios is foreseen for one of the next versions.

Stack-Capability Interconnection

The integration of several VMs of a heterogeneous set of hypervisors into a nested stack requires some compatibility checks and prerequires partially some specific resources to be setup and available. Therefore within the the UnifiedSessionsManager the interconnection-attributes

- STACKCAP
  Offered stack capability

[1] Lists are foreseen to be implemented, not yet available

- STACKREQ
  Required stack capability

are defined. These attributes are concatenated attributes consisting of various elemnts, representing the environment as defined by the various session types. Each entry consists of the same generic information, similar to the common standards of version information for OpenSource:

<name>-<version>-<platform>

The participating plugins, managing a specific session type, implement each the connectors for the upper and lower sessions plugin.

The stack capability is a mainly static information, which is inherent to the actually installed software component. Some variations may occur, when e.g. various kernels for various hypervisors are booted, thus varying the actualy stack-capability. The same may occur for a hypervisor, when several versions may be started as required.

Virtual-Hardware-Capability Interconnection

In addition to the Stack-Capability the Hardware-Capability supports some static but mainly dynamic parameters, which could be varied administratively and partially are influenced by a single or even multiple VMs.

An obvious capability is for example in case of QEMU the emulated CPU and supported motherboard/embedded-platform. Thus when starting an ARM-Version of debian, a QEMU-VM may be required, which(as requlary does) supports the ARM9-CPU and the required eval-board. Additionaly criteria may occur, when the actual load and the basic CPU-Frequency is taken into account, which may constrain the applicable machines.

The number of VMs running on a specific machine might be considered by the offered hardware-capability and the hardware-requirements. The number of CPUs/Cores and the actually present physical RAM are the first and obvious influencing parameter for this attributes. Additionally the "/home" devices, if local and physical, will be recognized. The usage of some metrics e.g. by a simple "dd-measurement" will be implemented as draft balancing criteria too. These sub-attributes are stored in the same manner as the stack parameters within the attributes

- HWCAP
  Offered hardware capability
- HWREQ
  Required hardware capability

The first version supports some basic checks only, more sophisticated evaluation and distribution may follow.

Access Permissions

The stacked access and the resulting "Propagation Scopes" impliy some advanced management of access permissions for several reasons. This is due too the two-step required acces, the first to enumerate potential upper peers by means of the hypervisor, the second to actually access the the upper peers by their native GuestOS. Thus two points-of-authorization has to grant access.

The scenario becomes somewhat complex, when it is taken into account, that in a stacked operations each peer might have it's specific user to be accessed, and additionally some specifics are required for an inherent mandatory operational mode when starting nested stacks. The most obvious case might occur, when native user-permissions only are choosen for Xen access, which requires root-permissions for starting a DomU. Of course, sudo and/or ksu has to be actually used here(and is checked by ctys). Anyhow, some login account variations might and do occur within a single "column of a stack". Another requirement is the neccessary polling of Native-Accessibility of the GuestOS, which is naturally not operated synchronously. Therefore the two functions "waitForPing" and "waitForSSHPing" are defined, where the latter requires native access by granted permissions.

The recommended usage is the application of NFS and SSO, when no security critical data has to be provided over the network. For advanced security requirements AFS or at least a newer version of NFS should be used. Alternatively parts of the filesystem could be defined by User-Space-Filesystems based on SSH for example.

The most advanced common Inter-Layer-Authorization could be implemented by the combination of Kerberos, OpenSSH, LDAP, NFS, and Automount. Variations with local and alternative networked filesytems, and without LDAP could be applied.

The general advantage of an SSO based access with a central filesystem is the simplicity of loadbalancing, when decisions for the <execution-target> and/or resource requirements related to the <action-target> has to be made. Particularly efforts for the physical relocation of a VM could be avoided, du it's system based distribution by the networked filesystem.

The ACTIONs supporting "Peer" mode and "Auto-Stack" mode provide an optional parameter to define individual users and credentials for an appropriate acces by usage of alternative accounts on the specific <action-target>. The access to the <execution-target> is performed as usual.

It is recommended to define a common set of users for access to generic purposes such as LIST, which could be commonly used. Additionally a specific case might occur, when a virtual bridge has to be created on the fly by networked access (Section 18.9 'ctys-setupVDE' on page 356), where during a required short-offline period some acces to stored executables is required. These ACTIONS should be performed by local-only users.

A second aspect is the authorization, where a local-only facility such as sudo should be used in that case, alternatively a cached approach could be utilized.

Verified examples for the application of VMSTACKs are available in Section 28 'VM Stacks' on page 559 .

Chapter 9

CTYS-Nameservices

9.1 Basics

The operations of the UnifiedSessionsManager utilize flat and stacked VMs, where a stacked VM is modeled as a nested stack of VMs, each level basically encapsulating it's upper layer VMs, particularly when these are offline. Thus the startup of an entity within a stack, where parts of the founding lower layer VMs has to be implicitly started before the destination, requires some sophisticated information of the pre-required stack structure to be utilized.

The next additional requirement results from the common usage of configuration files as the controlling data - controller-entity - to be addressed in a distributed environment. The following figure depicts the various configuration entries for an example with 3 Stack-Layers.

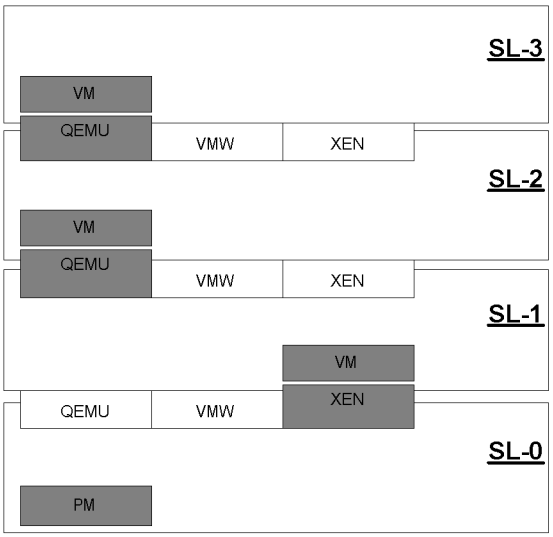

Figure 9.1: Stack-Controller Data

For additional information on formalized specification of VM-Stacks refer to Section 8.2.2 'Specification of VM Stacks' on page 88 .

The first layer SL-0 is consequently defined as the lowest layer of the VM-Stack, representing the founding physical machine PM, even though it is exceptional in it's concept, because of missing underlying peers. Thus the PM is represented by the PM configuration file only, and could be instanciated by dynamically utilising the remote Wake-On-LAN(WoL) via a relay for transmission of so called Magic-Packets(TM).

The remaining Stack-Layers SL-1, SL2, and SL-3 are nested within a VM-Stack, where each requires a two-folded view for nested and recursive operations.

The first step of operations represented by the initial controller data is stored within the configuration file specific for the actual hypervisor, in case of SL-1 from Figure:9.3 this is the Xen-Hypervisor with a configuration file for the DomU. The configuration file defines all essential attributes required for configuration of the virtual hardware, including it's peripherals, and some additional framework information. It particularly defines implicitly, by it's accessibility only, the locations, where the represented VM could be instanciated. This is, even though it is not an neccessarily an explicit at-

tribute, surely one of the most important properties of the VM. The accesibility, accompanied by more specialized attributes, is heavily instrumented within the UnifiedSessionsManager by location and distribution algorithms, including balancing approaches. It is of course the first-step recognition pre-requirement for the VM scanner and database generation tool ctys-vdbgen . Consequently, each <execution-target> is represented for the start of a VM by it's set of available and for the individual user accessible VM configuration data. The data could be utilized either from within one of the various caches or by a simple file scan based on the standard "find" command. An overview of the collection, and distributed and centralized processing components for the VM and PM configuration data is depicted in Figure:9.3 on page:116.

Several templates and generation examples for specific VM configuration files are provided within the templates and examples subdirectories installed within "$HOME/ctys". Some additional atributes may be applied to seperate additional configuration files or embedded within the native configuration file of the hyervisor(refer to Section 33.7.2 'Configuration File Variants' on page 625).

The second required view for additional consistency checks is the embedded PM/VM file within the GuestOS, which represents the OS itself and has to be generated by the utility "ctys-genmconf" .

Therefore some extended nameservice information is required, which for example contains information about hypervisors and the GuestOSs. These has to be frequently completed by the physical locality - PM, where a VM has to be activated. Even though the filesystem could be scanned for this information for each access, the caching of this data results to a perfomance gain on address-resolution by factors of at least 10 and more, frequently by a factor of 100 and more.

For pure dynamic plugins, which frequently do not have any stored configuration data, no configuration information is included the prefetched cacheDB.

The required hosting data of the GuestOs, namely a PM and/or VM, could be cached with the same rules applied as for configuration data of the hypervisor for the containing virtual hardware. This is supported due to a common file and record format for the ENUMERATE action, but also for runtime information by LIST action.

The operation of automatic startup of complete multi-level VM-Stacks requires some specific checks for basic consistency. This mandatory requirements contain obviously the compatibility of processor architecture, which could be evaluated statically. Some dynamic requirements as pure dynamic and/or semi-dynamic requirement occur. One example for this is the so called VRAM alocation. This could be pre-checked due to the several pre-configured amount of actually allocated VRAM, which is the maximum. Additionally the actual load of VMs within each stack entity has to be checked for teh actually reserved and/or remaining VRAM resource, which is influenced by the current running instances. Another obvious requirement is the physical consistency of the actual running instances.

A common scenario might be the case, where one instance is already running, probably an intermediate instance of the newly requested VMSTACK session. The requested stack thus could not be created straight forward, even though no additional constraint may exist. Several solutions scenarios may result of this situation, where some will be implemented within one of the next versions.

The current version supports for VMSTACK groups a fresh start only, thus no previous running instances of the stack memebers is accepted. This is also true, when the existing entities form a valid lower and therefore expandable subtree. In this case a VMSTACK for the remaining part, or each upper stack element has to be started seperately.

The resulting requirement of the several possible and of course partly very common scenarios lead quickly to the prerequirement of a completely populated cacheDB, containing each participant with it's basic capabilities. Additional dynamic checks are to be applied in order to detect and locate ACTIVE parts of the VMSTACK. Therefore the availability of the required data at all and it's visibility for the current state of pre-startup has to be considered.

The following figure expands the previous example of Figure: 9.1 with the actual state dependant visibility of the required static configuration data.

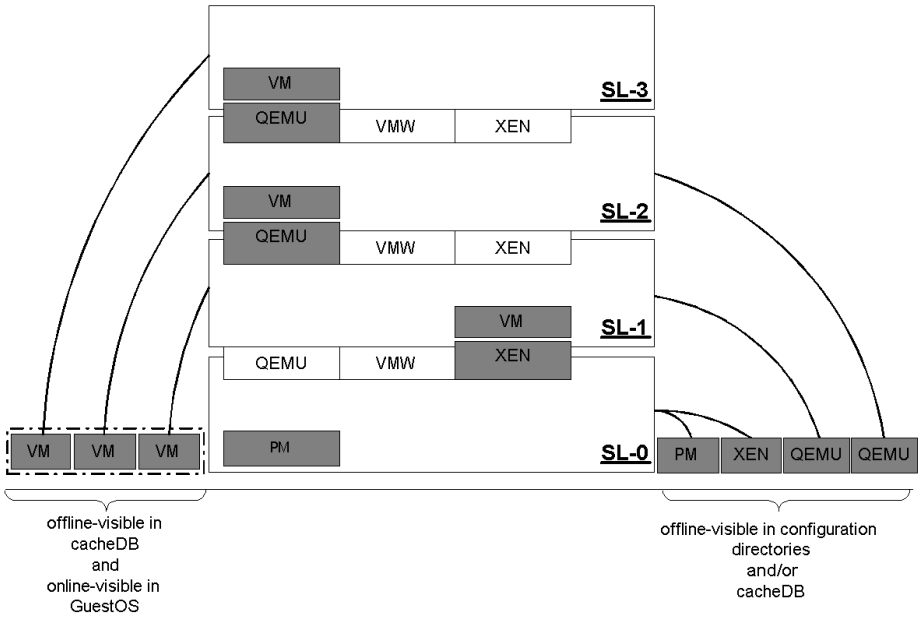

Figure 9.2: Stack-Controller Data Visibility

As shown, the inherent data of the GuestOS within the upper layers of the VM-Stack may not be visible by default as implemented within the current version. Therefore this version requires an initial pre-scan of a running instance in order to detect it's inherent available and accessible VMs.

Therefore the UnifiedSessionsManager implements various name-service measures in order to implement the required features.

- ctys-dnsutil
- ctys-extractARPlst
- ctys-extractMAClst
- ctys-genmconf
- ctys-macmap
- ctys-smbutil
- ctys-vdbgen
- ctys-vping

The TCP/IP data is integrated into the ctys-nameservice as IP from DNS and DHCP data. In a second step this is used in an extended manner to the /etc/ethers-approach and 0supported by it's own toolset.

Current support of IPv4 will be extended to IPv6.

Additionally to the basic user interface some caching strategies with related CLI options are supported.

REMARK:

The caching of nameservice ("-c") data is not related to the caching of payload data for distributed operations ("-C") . The data of distributed operations has to be cached for example when overall post-processing(like sort) is required. It could lead to a performance gain for repetitive operations too.

The caching of nameservice data is independent from this and is triggered by a dramatic performance gain and mandatory requirement for pre-startup structural knowledge of VM stacks.

This could be suppressed, when an actual file-scan for configuration files is requested.

The second and mandatory requirement results from the stacked operations and therefore in case of CREATE the required addressing of hidden, or "encapsulated" VMs within an upper VMs of the addressed stack. For the initial addressing in order to implicitly startup the offline base-parts of a stack, the knowledge of the whole path of address is mandatory, therefore this could be automated only by the presence of a sophisticated nameservice.

ctys therefore offers multiple distributed and local nameservices, which address the range from a single stand-alone machine to an extended enterprise environment.

9.2 Runtime Components

The components of the nameservice are structured as depicted within Figure:9.3 on page:116 based on the ENUMERATE and LIST action.

The main difference concerning Nameservices could be defined by the distinction of plugins with static administered and stored configuration data, and the dynamic plugins without static data. The nameservice components handle the static data based on ENUMERATE calls with some additional mapping, and with the LIST call for actual runtime states. The dynamic systems are handled by data from LIST action only. The temporarily user assigned LABEL is managed by dynamic runtime storage only.

The ENUMERATE action is utilized for interactive user queries and by internal queries for handling and caching of configuration data of VMs and PMs. Therefore the filesystems of enumerated targets are scanned by the call ctys-vdbgen for stored VMs. For detected VMs parts of the configuration data is correlated with network data extracted from the DHCP and DNS services and stored within a caching database.

The ctys itself utilizes ctys-vhost as first trial for data queries from local cache, when no cached data is available the target machine is scanned for VM configuration data.

The names of the depicted tools are similar to their functional equivalent UNIX utilities, thus the provided functionality is.

One exception is here the ctys-vhost utility, which manages and utilizes the whole caching database.

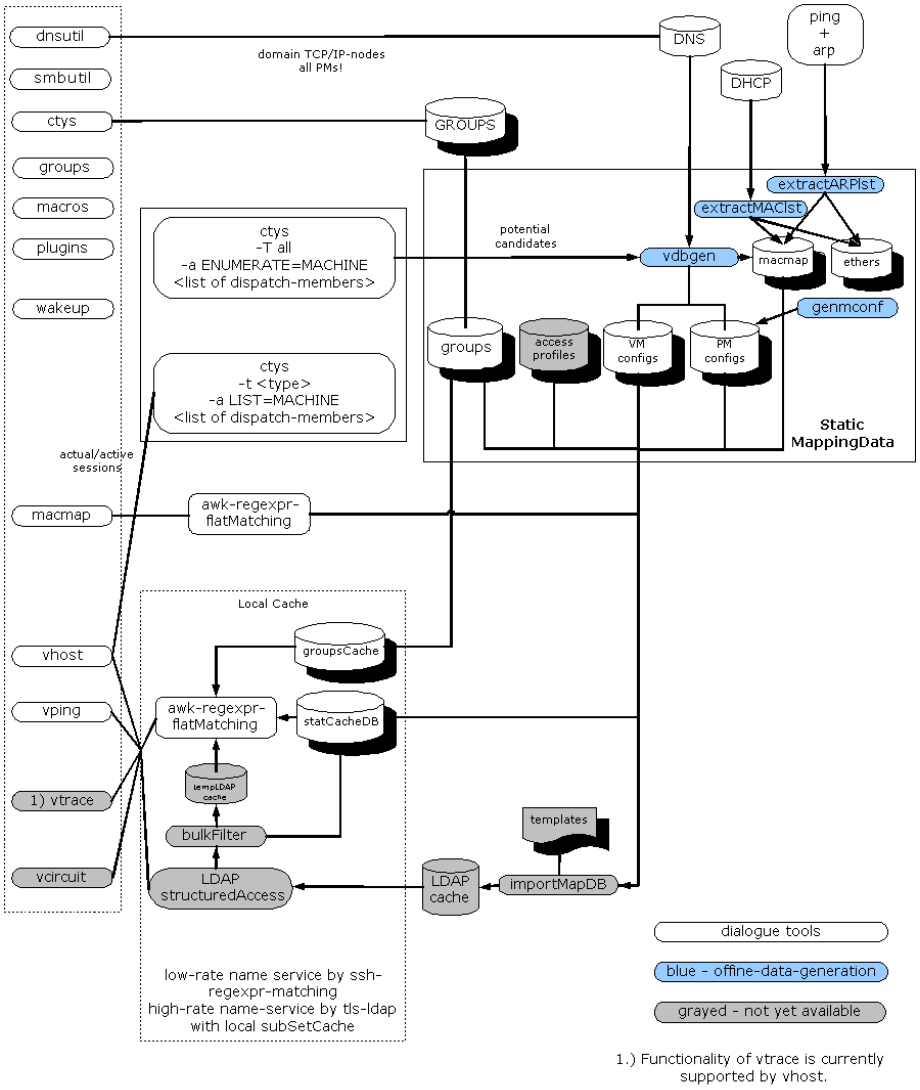

Figure 9.3: Nameservice components

The founding of the runtime database with the runtime tools by themself simplifies both, the runtime data management by the user and the internal data structures to be implemented, significantly.

The significant data management actions ENUMERATE and LIST provide the keyword MACHINE for output of a raw data format, which is the internal record format and the export/import format for data exchange with external tools. The data simply could be imported into any database or stylesheet, modified manually and re-imported into ctys again. Therefore the relevant tools support

the keywords TITLE and TITLEIDX for display of the actual names and canonical indexes of each field. The canonical index provided is the required field position when displaying data by usage of generic tables and the relevant position for custom user scripts, e.g. based on awk.

The cache database is organised as a file database where each entity is stored within one line constituting a complete record from a ctys-call. This has particularly the advantage, that for the typical search requests for the mapping of one field to another an easy to define awk-regexpr could be used by the caller. This of course could be a simple string. Substrings representing partial information of an item are matched by all superposing strings which are longer than the search string. Therefore the listing of members of a subnet, or members with MAC-addresses from a single supplier is almost a trivial query with given interface. The listing of contained elements within a container of a lower level is as trivial too, by just supplying the container's id as search string. This is particularly very handy in comparison to the required full-path IDs for VMs of some tool.

When using the ctys actions as ENUMERATE, the produced output is literally ready to be used within the ctys-nameservice file-database. The output is just redirected by the wrapper tools into the runtime files. This opens for the user particularly the asset, that different local cache databases of several executions groups could be simply created by selecting members for a call. The native ctys-call is encapsulated by ctys-vdbgen with a little overhead in order to manage combined calls and some common databased-paths for usage by the relying runtime support tools. Therefore ctys-vdbgen should be used instead of rudimentary native calls, even though is possible without any required data mediation.

A second tool called ctys-extractMAClst generates a mapping database "macmap.fdb" from a dhcpd.conf file with three column information, mapping host names, TCP/IP-addresses and MAC-addresses. Just to mention for completeness, this tool also supports the generation of "/etc/ethers" from a "dhcpd.conf". Alternatively the tool ctys-extractARPlst could be used for the same issue, but with dynamic polling.

These both databases will be utilized by the runtime tools in order to generate complete mapping information when required.

In current implementation, which is the first system to be stabelized and probably discussed somewhat, the mapping data is stored individually within a configuration database in the home directory of each user, which is $HOME/.ctys/db.

9.2.1 Distributed Nameservice - CacheDB

As mentioned the current version stores it's data in file databases, with one record each line. The fields are seperated by semicolons, so these files could be viewed and inspected with almost any spreadsheet-tool.

Performance is for small and medium networks quite good, where medium might be up to some thousand entities. This depends on the actual machine of course.

The remote data will be fetched and collected by "ctys -a ENUMERATE..." calls, what could be somewhat time consuming, but is cached locally within one call, and could be done persistently too when set.

The build of the local cache on each node is performed by three major steps.

1. ctys-extractMAClst or ctys-extractARPlst
   Sets up a MAC-IP mapping database.

2. ctys-vdbgen
   Collects configuration data from a given list of hosts/accounts.

3. ctys-vhost
   Pre-Converts data from a first-level cached data-format into a runtime format, where almost only some IP conversion and precombination of groups with each members data is performed.

The components are as given in Figure 9.4.

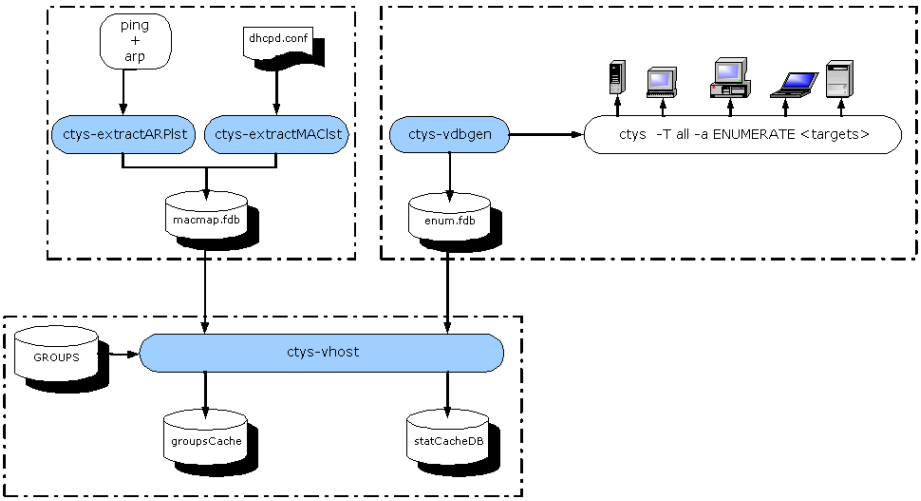

Figure 9.4: Cache Generation

The data could be cached in local databases on multiple nodes. It will be decided at runtime, which cache should be used.

> The default behaviour is to try the local cache on the caller's site first, when no cache-hit results, the remote site's cache is tried next. If again no chache-hit occurs or no cache is available, than in case of a "configuration based" plugin the remote file system is scanned for the appropriate configuration data. If a config-file is found, the related VM is started.

The default behaviour "-c BOTH" could be altered to "-c LOCAL" or
"-c REMOTE". The final scan of the filesystem could be suppressed by the "ONLY" suboption.

For pure dynamic plugins such as CLI, X11, and VNC no cache will be used, thus no filesystem scan is required too.

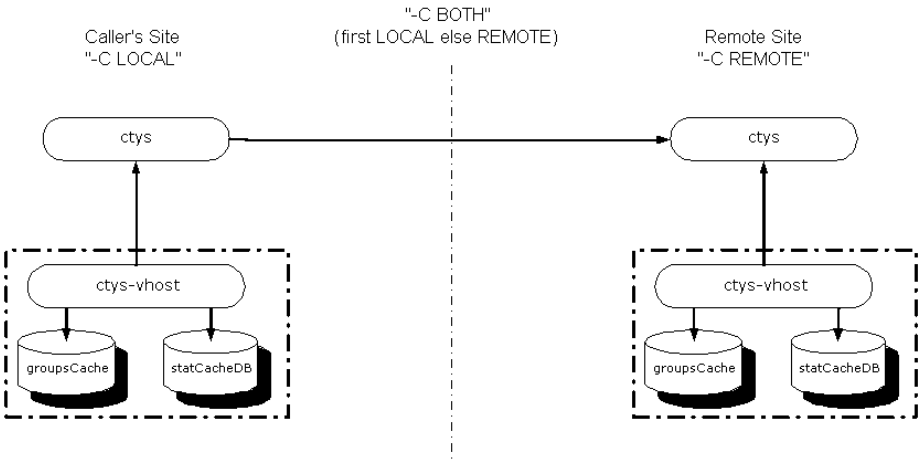

Figure 9.5: Distributed Caches

When LOCAL cacheDB is selected to be used, either by LOCAL or by BOTH, the reelvant data has to be considered on the callers site and additionally to be transformed to the remote site. Therefore the technical implementation is based on the remote call based on SSH and the ctys-interface. The local cache is resolved to a fully qualified <machine-address> with all of it's actually present parts, and inserted as the first argument of the relevant ACTION for the current job. This is performed on the current task to be activated, when working out the stack of the internal schedular, just before the local/remote task is finally executed.

The basic design and resulting implementation for the work out of the actual values of the locally pre-fetched cache is based on the implementation of options evaluation with permitting repetition of the same option/suboption and the "Last-Win" philosophy for multiple occurances. Resulting from this user supplied parts will superpose elements from the cache and may lead to desired and/or unintended deviation from the local contents of the cacheDB. Anyhow, the iput used for evaluation of the <machine-address> from the cacheDB should be in sync.

The main advantage of this approach is the efficiency in implementation, particularly requirering less or no additional application specific implementation, when design rules and coding patterns are applied. The approach is inherently generic and could be applied to any ACTION requiring one <machine-address>, as syntac-

tically allowed in current version. Potential multiple occurances of <machine-address> will be added later, but could be designed as an neatless extension.

9.2.2 Network LDAP-Access

Even though the performance of the tools seem to be perfectly alright, the centralized management and distribution of network information data would be preferred. Therefore one of the next versions is foreseen to rely on a LDAP implementation.

9.2.3 Application Range and Limits

The nameservice utilities manage and provide mapping of ctys-names for VM-Addressing to TCP/IP addresses.

The following restrictions should be recognized.

- The ctys toolset handles by TCP/IP or better by OpenSSH interconnected entities. Thus HostOnly networks are not supported.
- The VMs like Xen and VMware rely as one of their most important identifier on the MAC-Id, particularly when DHCP is used in highly dynamic networks, where the VMs could be seen as roaming "virtual-devices".
- ctys does handle DHCP based mapping, but does not support address-pools. So static administered MAC-addresses could be used for selection by ctys tools only. Anyhow, as long as this is not required, ctys perfectly cooperates with dynamically assign TCP/IP-addresses.

9.3 Required Namebinding

A session is defined and accessed by it's name binding. This is actually different for almost each integrated type of sessions a.k.a. plug-ins within their native namespaces. Therefore the following unifying name binding has been defined, which still supports the several specific namings as an additional facility, to be used for some required specific use-cases and/or call of specific tools.

The ctys modules which are implementing the namebinding particularly supports conversion between the several naming attributes,

thus a common interworking could be setup.

In addition a common nameservice is defined, which offers a binding and cross-resolution between the different plugins. Distribution and transparent caching, including security aspects is included.

9.3.1 Integration of PMs, VMs, and HOSTs

The integration of the supported categories PM, VM, and HOST is the essential advantage of the implemented namebinding. Therefore a superset is defined as protocol entity and implemented within the several plugins. Basic functions with plugin specific call interfaces are supported as a common library.

The main advantage results from the combination of the various IDs addressing several entities of different categories and types into a common and centralized database. This address-pool contains identifying data from within multiple OSI/ISO-protocol layers, as well as from multiple application domains. The common access with additional dynamic runtime data and the combination into runtime profiles for service-locator-decisions offers broad application of the nameservice.

The actual execution entity ctys-vdbgen for generation of static data and the ctys-vhost runtime interface to nameservice could be easily extended individually for implementing specific policy based nameservices. The supplied basic version offers a functionality for combination of dynamic and static data and application of several filters on them.

The different plugins of the defined categories need not necessarily to implement all elements of the common naming schema, e.g. the type HOSTs has normally not a configuration file, and the IDs are generated dynamically by the executing OS. Whereas the VMs have a configuration file with almost the whole set of relevant runtime information founding the uniqueness of a instanciated Vm under operations.

Plugins also can define additional elements for their own namespace, but which is not allowed to replace other mechanisms. For example the VNC plugin implements a "BULKCNT" attribute, which supports a facility to start the given number of sessions at once and sharing one encrypted tunnel for DISPLAYFORWARDING. This

feature is particularly used for testing purposes.

The big unifying element is the commonly supported LABEL, which has slightly different semantics within the different namespaces, but supports in any case it's main intention of a common access alias. The LABEL could be used as any other namebinding element and supports the full scope of functionality for addressing entities. The only drawback is the responsibility of the caller for it's assignment and therefore it's uniqueness.

Other IDs might be unambiguous by default but could be tampered by in-place-backups or multiple mount of shared file systems, additionally by user-errors of course.

So the challange of ambiguity will remain in any case, but could be coped easily with some discipline. The in-place-backups will be treated by using the first attempt in most cases, where multiple results could occur.

9.3.2 Basics of Name Bindings

The namebinding and therefore it's evaluation to resulting entities is in many cases not explicitly defined to match completely by algorithmic decryption. This results from the following characteristics, which offer the real advantage of the given schema.

- The address elements are foreseen to be used as search pattern for resulting entries which will suffice given keywords as attributes within specific instance contexts. E.g. it is possible to search a configuration file representing a VM, which is localized within a given subtree, and which contains a specified UUID. The resulting query by ctys-vhost will give the selected item, TCP/IP-address or resolved filepath with hosting PM.
- The second reason for a stepwise evaluation of given namebindings is the distributed nature of the addressed entities. This means, that an immediate evaluation of the resulting values on the callers site is in much of the cases simply impossible due to the remote-only availability of the required data.
- Completely dynamically allocated entities will have just a few static inherent information, like their generic call name for execution, whereas almost any additional identifier related to the running instances will be generated by the hosting container,

e,g, the guestOS within a PM or VM. In that case the LABEL could be the only reliable identifier for a specific session.

An internal interface is defined, which evaluates the given parameters with multiple calls of an plugin interface by a central dispatcher. The interface is mainly based on dispatching tasks by it's given type("-t/-T") and ACTION("-a") and passing pre-analysed and assigned generic command line arguments and resolved permutations to the specific plugin. The information for each task is called her message.

The main dispatcher distinguished between two types of messages to be passed to the plugins, one type are generic messages related to multiple plugin types by definition, these will be handeled centrally by simply calling specific format converters. Messages or better actions of this type are LIST, ENUMERATE, SHOW, and INFO.

The plugin type specific messages are passed to the affected plugin with the action type and an internal action, which is actually a sub-action. The sub-actions are CHECKPARAM and ACTION, where the CHECKPARAM by interface convention just implements a local check of parameters as possible on the caller's site, whereas the ACTION message does the final checks at the site of execution target and performs the requested actions.

The expansion of names related to the transfer of action to the execution site, e.g. when several PMs are selected for one USER differing from the caller's USER-ID, each host from the given set of addresses will be combined with the defined user and the remote execution is performed.

Another issue arises from the usage of addresses resulting from namebinding with context specific options, which apply as a superposition of the globally set call-options for the specific target only, this could be given for each target specifically.

Ctys handles all of this cases correctly.

9.4 Group-Targets

ctys supports the usage of groups, which are actually file names containing a list of hosts to be handeled together. This list could be any user defined assembly of single-targets by any criteria, to be

executed together as a group.

Technically the group name within the CLI is simply replaced by the set of host names from the group file. Additionaly context options for each group are permutated for the resulting set of hosts from the group. Group elements may be in current version bare target names, without stored context options.

Extended groups with specific pre-customized functionality are available, which is controlled by a syntactical keyword. Additional information is available in Section 13.3 'Hosts, Groups, VMStacks and Sub-Tasks' on page 161 .

Due to the previous expansion of given groups, the group name will hide any actual target with that name from the CLI, whereas this name could be reused within a group file. Groups are could be nested by "#include" statement, multiple groups could be listed, seperated by commas. The group statement could be present more than once for each include-statement:

```
\#include <group>[,group[,...]]
```

The '#' has to be the first character of the line followed immediately by the literal keyword 'include'.

Group files are stored and searched by default within the directory "$HOME/.ctys/groups" and "$MACONFPATH/groups", but multiple search paths could be provided by the environment variable. The syntax is analogous to PATH-syntax.

```
CTYS_GROUPS_PATH=<absolute-path>[:<....>[:<...]]
```

The first match will be used. The file itself could be padded with comments and empty lines, which are ignored when evaluating it. The default paths are prepended to the CTYS_GROUPS_PATH during init of ctys.

Part II

Software Design

Chapter 10

Software Architecture

10.1 Basic Modular Design

The architecture and design of the UnifiedSessionsManager is targeted to be capable for integration of any third party tool. Therefore a plugin systems with dynamic and fully automatic load and operational state evaluation is designed and implemented.

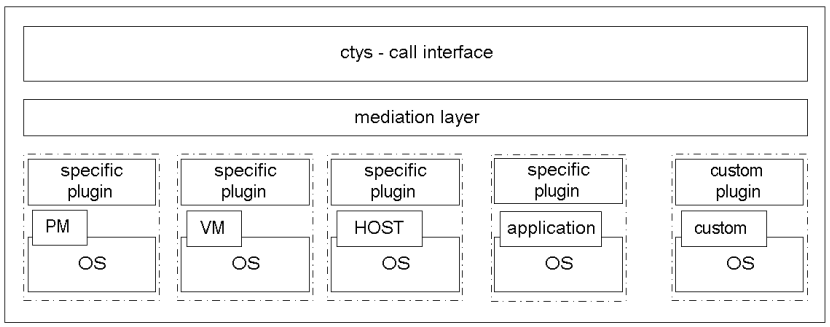

Figure 10.1: ctys Software Layers

One main driving argument was the required amount of memory space for the bash modules, including their embedded comments, which originally exhausted the mainly static defined bash memory. Particularly some older versions on UNIX variants lack the ability of means for definition of the available memory size within shell calls. The decision was made not to "strip" the code, but load parts actually required for the current task only, very similar to the so called "Shared Objects" or "Shared Libraries". The dynamic on demand load of plugins reduces the resulting memory requirement significantly.

The usage of the bash has some limitations of course, but it's simplicity and common applicability by the targeted audience made the decision for it, at least for the first version. So almost anybody should be able to customize and extend the UnifiedSessionsManager as required. It is prooven to be possible to implement a new VM or custom application within a few days completely, including required tests. This is possible because of the presence of comprising but restricted number of standard interfaces only and implemented plugins as reusable patterns to be combined. The support environment with the included library modules of the framework accomplishes the task.

The unification of the call interface requires the restriction of the interface to essential actions only. Thus specific features are not supported by ctys tools, but could be executed within the HOSTs packages as a native call. So even specific and prorietary features could be used with full advance of common generic features such as stacked VMs addressing, connection encryption, and GUI management.

The usage of a dynamic scan and load mechanism for available plugins which detects present plugins and sets their operational state appropriately, has the advantage of providing a simple configuration opportunity for any user. A simple remove or copy of a plugin subdirectory combined with eventually minor configuration is all required to be done.

A similar approach of scanning for valid configuration files of the managed hypervisors is applied, when the available and operable VMs for each plugin are enumerated. The specific configuration for each plugin is selectively detected by a so called match-filter, which copes the analysis of the initial non-ctys-aware configuration file too.

The execution of ctys is splitted into a client and a server component, where both components are identical, but perform different functions in each context. It should be mentioned here, that the compatibility of remote calls between client and server component of ctys will be checked by the hard coded version string of the "ctys" call. The plugins and library modules are not checked, but just trusted to be suitable to the overall ctys version. This is for complete installations and updates true, but could be undermined when installing or removing components only. Particularly the standard

HOSTs component VNC could have serious impact on the VMs XEN and QEMU, when used as console.

10.2 Development Environment

As a draft pre-remark it should be mentioned, that the context help supports the listing of functions and their headers. Thus supports an embedded online help system.

Additional documentation will follow.

ffs.

10.2.1 Some basics on "bash"

The benefit of the bash is obviously it's widely and easy applicability due to the available knowledge of the targeted users.

The usability of ctys within shell scripts, offering a namebinding for login configuration of complex user interfaces and enhanced desktops is another argument.

But there are some drawbacks, which do not eas life very much. The main challange is the missing of structured data structures and associative arrays. Even though this could be partly emulated, the decision to a remarkable number of global variables was made. Arrays are used for task management only, which would be utilised by incremental access only.

Another challange is given due to the limited namespace capabilities and inheritance behaviour within nested function definitions. The definition of nested calls for loops has to be checked frequently for overlapping or better local-only definition of index variables.

But anyhow, the lack of an implicit namespace was the feature what made the design of a function based dynamic-source-ing environment possible at all. So, really could be said "it's a feature, not a bug".

"source-ing"

The most important feature used within the UnifiedSessionsManager is the source of components. This is developed within ctys

tools to be used as similar approach to shared libraries of the compiler environment.

Therefore almost the whole scripts of the ctys tools are just plugged together by using a load mechanism based on "hook" convention.

Using bash-modules as self-configuring dynamic objects

ffs.

10.2.2 Component Framework

ffs.

Hook - The Extendibility Interface

The interface to load a component dynamically is based on some minor but important naming convention and a basic set of interfaces. This includes a level based set of initialization calls and propagation of these.

ffs.

Static Load of Modules

ffs.

Dynamic OnDemand Load of Modules

ffs.

Operational States

The available plugins are utilized and managed by usage of their integral state variables. Particularly the operational state defines the accessibility of the features provided by individual plugins.

The usage of sets of plugins could be triggered by explicitly called CLI options "-t" for specific targets, or by the pre-load option "-T". In case of generic actions like LIST, ENUMERATE, SHOW, and INFO, ctys will parse all loaded and available plugins for representation of statistical or bulk results. This will lead frequently to errors, when some prerequisites of individual plugins are not met for any reason.

The basic issue when implementing a plugin and using it at run time is the availability and operability of the prerequisite third-party software components. This could involve the detection of the appropriate software call interface as well, as the validation of it's successful intialization. This becomes cumbersome quickly, when managing it manually in a highly volatile test and development environment, where multiple kernels and multiple hypervisors, additionally multiple OSs are used.

The continous installation and deinstallation of appropriate plugins seems not to be available approach.

Therefore the operational state of each plugin is introduced. Any available plugin could and will be installed now and independently for each call it will be detected whether the operational prerequisites are met and the state will be set appropriately.

The state variable for a plugin will be dynamically determined based on the array of runtime detected and loaded plugins. Therefore in the entry-hook of each plugin a variable with the following naming conventions has to be defined:

```
<plugin>_STATE
```

where <plugin> is the name of current plugin in UPPERCASE, same as the containing directory name. E.g.

```
XEN_STATE or VNC_STATE
```

The variable can be assigned one of the following states similar to the ITU-T definition for telecoms:

AVAILABLE(0)
: This is the implicit state of any found module, which is not handled actually. Or better to say any present plugin is registered in a list of available plugins, which represents it's presence. The default state is set to DISABLED, because this is the first

non-obvious "level" of state requiring management information exceeding it's pure presence recognition.

DISABLED(1)
This is the default state, which is stored in the runtime modules.

ENABLED(2)
This state is set, when during initialization of the module any prerequisite is met. Else the state remains DISABLED(1).

IDLE(3)
This state will not be utilized, due to the fact that all tools are One-Call Utilities, not daemons. So any ENABLED(2) plugin will be utilized to BUSY(4) or just remains as pre-checked, which is ENABLED(2).

This behaviour might change (but does not seem so).

BUSY(4)
This state will be managed just for statistical reasons when for ctys the "-v" option is choosen. This option displays after completion of execution the last state of ctys just before termination. Therefore it shows the actual state of the completed call, thus marks the actually used modules as BUSY.

REMARK: This feature is not yet finally implemented.

IGNORE-Flag - A stateless State

There is a special flag available, which could just be named indirectly a state, if so it would be similar to DISABLED. But this does interfere and not match with the used DISABLED state, which is the result of an at least partly failed initialization of a present and at least basic executable plugin.

In difference the IGNORE flag - which could be constrained by simple bash-conditionals - influences the loader and prevents the load of the plugin at all. This is due to configuration, for whatever reason.

So the IGNORED state is actually the "unavailable" state, avoiding execution failure during initialization on an unsupported platform e.g.

Commonly in the configuration file "ctys.conf" for each plugin a flag with the syntax

```
export <plugin-type>_IGNORE=1
```

could be set and will be recognized by the loader bootstrapping hooks. The <plugin-type> is in accordance to common ctys convention the literal name of the plugins own root directory.

The implementation as a bash variable which could be initialized by evaluating arbitrary constraints as simple shell scripts, offers particularly a smart means of machine dependent assembly of plugins. This works also perfectly when the identical file is commonly mounted by all of the execution targets as VMs and PMs. It is for example a common managed task seamless distinguishing between client and server locality for a Xen session and therefore cope the different runtime environments with it's completely different requirements. This is particularly the case for the CREATE and CANCEL action to be performed within the Dom0.

A detailed example is provided within Section 20 'ctys Setup' on page 415 .

Multi-OS Boot Environments

The usage of multi boot environments opens a bulk of issues to it's management. One is the installation of appropriate software components and not using parts, which might fail. Another might be the ongoing synchronization of updates for the system and the available plugins.

Therefore the decision for ctys was the introduction of the previously described state variable. The state of each plugin will be determined dynamically, so all available plugins could be installed at once. Even though e.g. the Xen plugin might not necessarily usable on a configured kernel for VMware.

This feature becomes very handy, when e.g. in case of Linux several kernels are installed for the usage of multiple hypervisors. E.g. VMware for building OpenBSD and Xen as paravirtualized on non-HW-VM for some number crunching on a partitioned Linux machine with some Linux-VMs.

Interface-Specification - Online-Help

ffs.

Interface-Specification-Rules

ffs.

Generic Context-Help for Shell-Scripts

ffs.

10.3 All about bash-Plugins and bash-Libraries

ffs.

10.3.1 The Differences

The main differences between libraries and plugins are the static load behaviour and fixed hardcoded load of libraries only. Therefore no init procedure despite the automatic processing of calls is performed. This is equivalent to the init-level 0 of plugins.

Another difference is the more conventional reason than technically, that libraries are designed as generic components to be used in any project. Whereas plugins are specifically designed in order tto support a unique project primarily.

Plugins are loaded static and/or dynamically. They could be just default initialized when required.

10.3.2 Libraries

ffs.

10.3.3 Plugins

Category CORE

This category offers basic features, which are generic and applicable to multiple specific plugins. Thus CORE plugins are very close to libraries, but are project specific and loaded automatically. Sub components could be loaded on demand by generic scanned top-level entries.

E.g. the CLI CORE component (which has nothing to do with the CLI plugin), handles the ctys tools specific options scan for a number of tools.

The GROUPS component handles the resolution of group names into host entities.

Category HOSTs

These category contains plugins to be just executed natively within a running OS. Therefore most of them serve as console clients for VMs.

Currently the plugins CLI, X11, and VNC are supported.

Category VMs

These are the bread and butter applications for stacked VMs. Each of them support at least one specific VM. Some support multiple variants, when these just do require minor variations only. When more specifics are required it is recommended to support another one seperately. This helps to reduce the required runtime resources as well as the reduces the maintaining efforts, even though some parts might be redundant.

The standard support is available for XEN and QEMU, and for the initially implemented VMW, which includes Server, Player, and Workstation.

Category PMs

The PMs plugins support the required functionality for handling of physical machines. This varies somewhat from VMs in various aspects.

The main difference for conceptual reasons might be within the CREATE method, what has to be executed for the initial "switch-on" of the PM on another machine. This breaks he basic command call structure as the only exception, where the action arguments "-a CREATE=.." contains the subparameter referring not to a contained subinstance, but even cross-over to another machine without any encryption. Which is required for the initial Wake-On-LAN packet.

OK, this could not really be seen as a security flaw and thus designed this way.

Another point is the automatic opening of a console session. This has to be performed in an exceptional structure too. This is due to the same reason as the initial WoL packet. The session could be opened from another machine when the execution-target to be waked up is SSH-accessible. Therefore a polling mechanism based on timeouts and trial-counters is implemented.

The stacking and therefore the state-propagation works similar to the VMs.

10.4 ctys Control and Data Flow

10.4.1 Distributed Controller

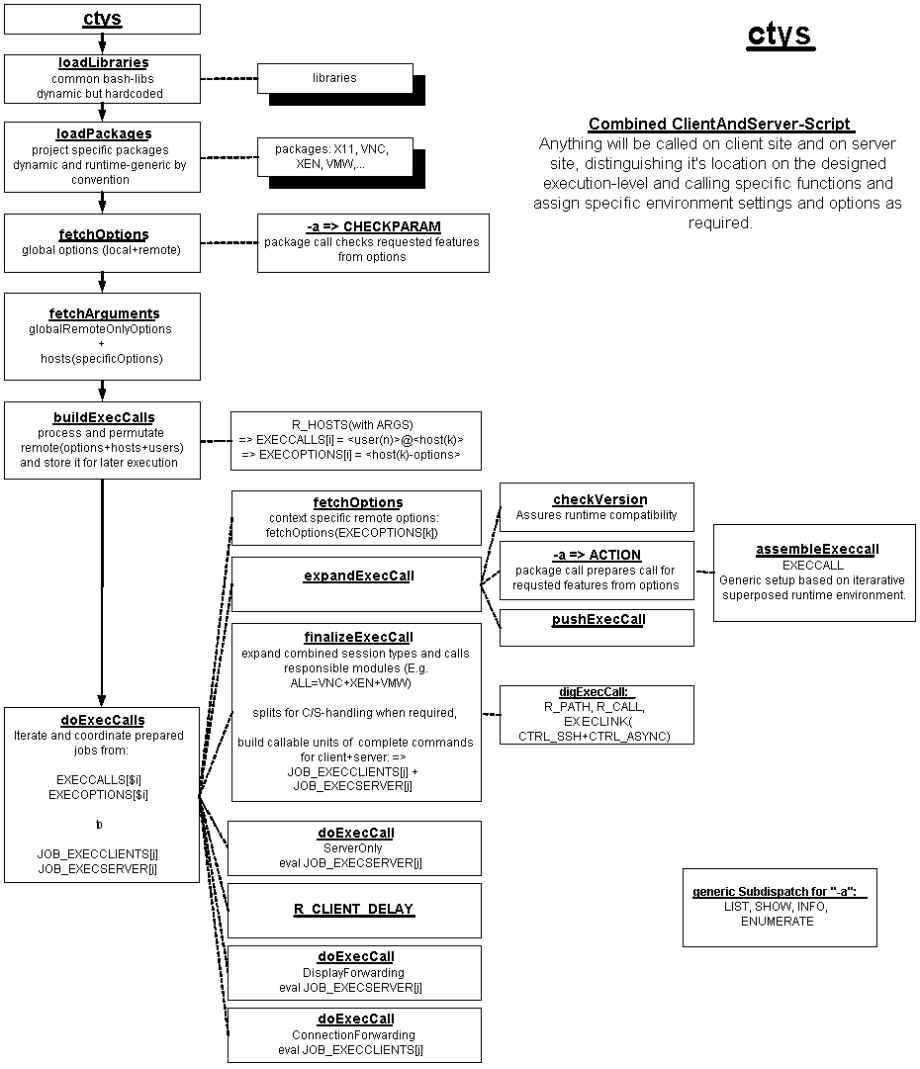

Figure 10.2: ctys Local Control Flow

10.4.2 Task Data

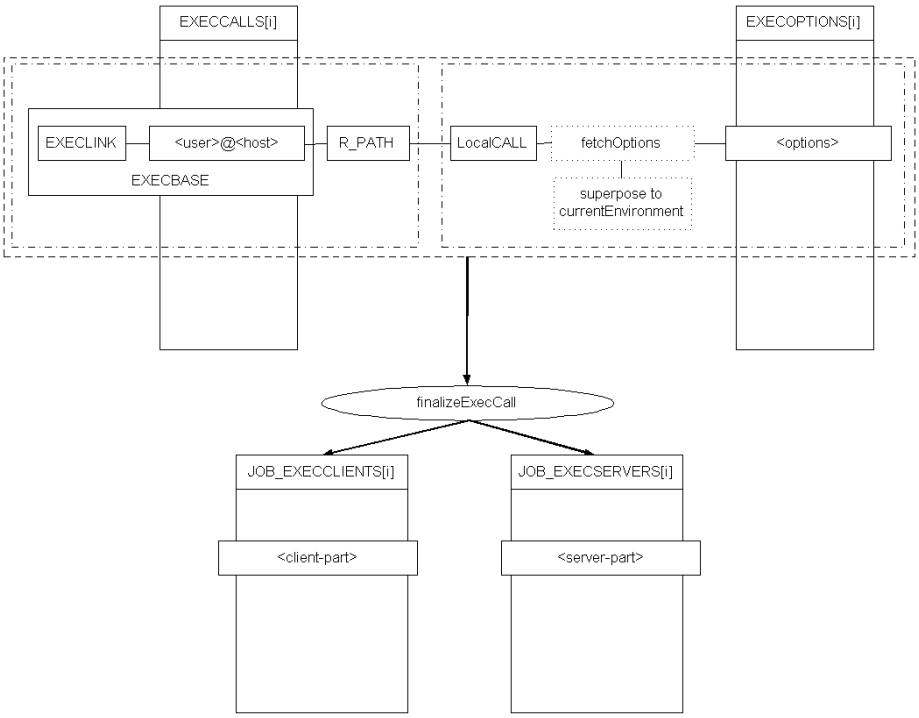

Figure 10.3: Task Data handled by the main dispatcher

10.4.3 Stack Interworking

Create Propagation - CREATE

Upward Propagation - CANCEL

Upward propagation is utilized for a controlled CANCEL of a stack with multiple levels of nested VMs. The demonstration in current version is implemented with VMware and XEN as bottom VMs running in a PM. The upper levels are implemented with QEMU as a CPU emulator without it's kernel module. Thus a nesting is supported without specific kernel involvement. The first release of stacks is tested on Linux bases only, even though any UNIX based platform might work.

The upward propagation during a CANCEL action performed on the sl-x (stack level-x) requires a successive upward walk through any contained stack and initiation of the appropriate resulting actions beginning on the topmost VMs. Thus the algorithm first detects it's upper tree and dispatches CANCEL request to each involved

instance. Once the whole tree is resolved, the top-level VMs begin independently to CANCEL their hosting instances. When all upper peers of a VM are CANCELed, than the intsance itself performs a CANCEL.

In case of a pure termination things are somewhat easy to implement, but e.g. in case of a RESET the only actually resetted instance is the first called instance. Due to simplicity in this version no implicit reset of upper parts of a stack is supported.

The involved components are depicted in the following figure.

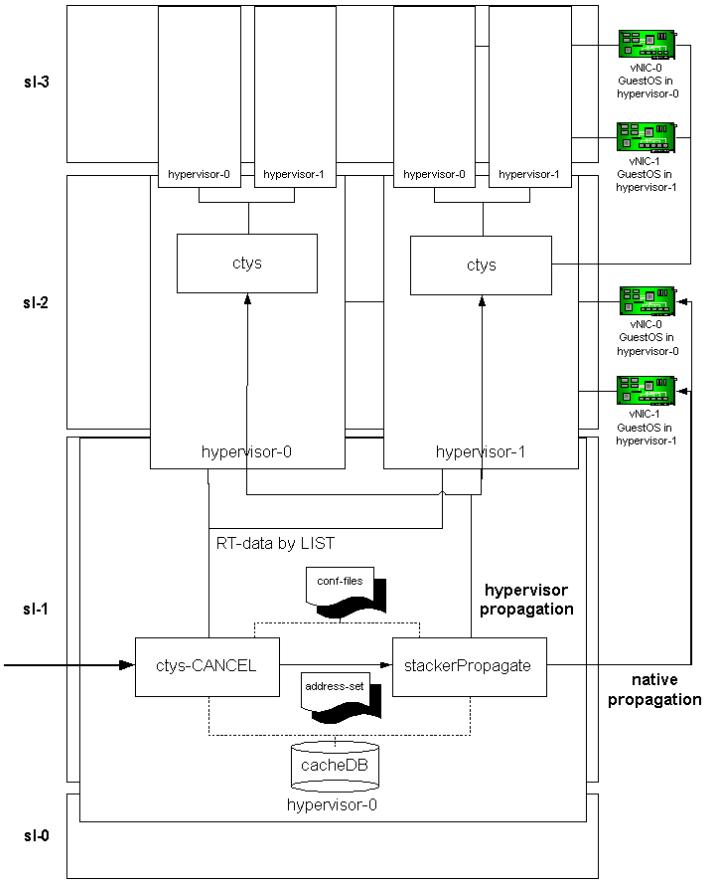

Figure 10.4: Nested Upward-Stackpropagation

The implemented control mechanisms are designed as a recursive three-stage algorithm.

1. native propagation
   If FORCE is not set, than the stackPropagate function is called

first. This function implements a recursive walk-upward call propagation of ctys with CANCEL action. The success and finalization of the called method is here monitored by a time-out value only. This again is designed due to simplicity and avoids sophisticated persistent state-control measures and implementation of controller deamon services. Also some specific advanced cases of handling states of upper parts are avoided.

The native propagation itself executes as a final approach a hypervisor call, when an instance within the stack is not accessible. This is required due to the pure "implied state-control" within the recursion through the stack. Thus any intermediate level instance without native access would be handeled by it's own hypervisor, instead of the containing. Within a proper setup of tightly integrated stack instances this case might only occur as an erroneous exception.

In distinction the the hypervisor propagation immediately accesses the hypervisor and just relies of it's and it's nested inherent hypervisors proper capabilites. Thus this final call is not an actualy redundancy.

2. hypevisor propagation
When the native propagation is finished the remaining instances are handeled by direct interworking with the hypervisor. This has advantages and drawbacks.

One dominant drawback is the required awareness of the hypervisor of it's contained GuestOS, and the support of adequate interworking tools for handling a simulated harware reset and switch-off combined with a proper shutdown. If lacking the situation is almost the same as abrubtly switchin off any physical UNIX machine without a previous shutdown, thus e.g. a missing sync.

The advantage is the uncomplicated implementation of a GuestOS independent interface for CANCEL. This is used e.g. for MS-Windows running in a VMware hypervisor with installed VMware-Tools.

The only available control by a predefined TIMEOUT requires a suboptimal setting of it's value in accordance to the worst-

case. Any partial shutdown will be forced to finalize immediately, when after the first trial due to reaching configured timeout a second step with immediate takeover of control by the hypervisor is entered.

3. self
   The SELF call is the final CANCEL of the host containing the initial call, which could be an instance at any level of a VM.

Downward Propagation

Chapter 11

Interface Design

11.1 Target-Platforms

11.2 Feature-Design

11.2.1 Communications Modes

The communications within the UnifiedSessionsManager between physical nodes is supported by OpenSSH only. Thus encrypted connections are supported only. This is established either by the X-forwarding mode, or the explicit setup of a port-forwarding channel. The two main cases to be distinguished are DISPLAYFORWARDING and CONNECTIONFORWARDING.
The other main difference is the location of clinet and the server peer, which are executed collalocated in case of DISPLAYFORWARDING but splitted onto different nodes in case of CONNECTIONFORWARDING.
There are just some specific technical driven chracteristics to be recognized.

SERVERONLY
: First of all, this case is intended to starts a remote server only, as far as supported by specific package, in headless mode. No specific communications will be addressed in this state. This mode will be primarily addressed as an internal starter mode.

  Some plugins do not support the headless mode for startup, but allow to cancel the client only for continued headless mode operations of the server.

LOCALONLY
: This is the most usual case for common application software

in proprietary single-user-only environments. This is not the focus of this tool, there might be better solutions, for that, if at all required. But a very similar case, the display forwarding is.

The following cases are the main application area of this tool, which focuses on massively distributed environments.

DISPLAYFORWARDING
: This is a frequently applied default mode for usage of modern architectures in a single user environment often within a standalone or file server level networked single user machine. It suits also good to servers with thin clients, where the display features do not need to be of enhanced graphics functionality. Almost the whole required resources are offloaded to the server here.

  The whole display will be forwarded to the clients machine by an underlying systems protocol, in this case the X11 protocol with display redirection will be used. This will be also applied, when a complete virtual desktop based on VNC and/or a virtual machine is started on the server. For security reasons and smart application only the OpenSSH package is supported.

CONNECTIONFORWARDING
: In this case the most of the graphics processing will be done native on the client site. Which utilizes on one hand protocols like X11 protocol of local application specific clients and in addition uses XClients to utilize the local XServer. With tools like VNC based on FBP the client will be executed in locally and redirected by port-forwarding via an encrypted SSH tunnel to the remote server.

The DISPLAYFORWARDING is compared to CONNECTIONFORWARDING the simpler case when using SSH with a VNC server. It is almost just required to set the "-X" flag and all the communications between the peers will be done automatically through that one channel only. The termination is by definition coupled to the session, so exiting the terminating application closes the session.

When using CONNECTIONFORWARDING the things become some more complicated. The decision here was made to using port forwarding literally only, but not in reverse direction in order to avoid a listening mode on clients side.

Due to some limits of actual implementations a channel-bundling is not supported by OpenSSH for X-forwarding when used with the "-L" option for an explicit forwarding tunnel. Particularly VNC binds each of it's DISPLAYs for each call to another listening port. Clients front-ends like VMware-Workstation works fine, but are used to unification in a common manner. So for this mode for each session a new SSH tunnel will be created dynamically in so called one-shot mode.

Another issue results in the management of multiple desktops on the clients display. When using multiple desktops with a supported tool like "wmctrl" particularly in an arbitrarily intermixed mode it would be preferably to set up several desktops in the background and switch them to the foreground as required. The restrictions here result from the limits and unreliability of handling windows and therefore placing them on arbitrary desktops(which would preferably be in the background of course). So the most reliable workaround as applied here seems to be grouping the windows desktop-wise and switching to the desktop when visualising them by default behaviour on the current. The usage of Proxy-XServer or any kind of ghost-desktops for preparation is currently ffs.

The support for this option depends on the session type and the current action. E.g. the LIST action to show the users sessions might be aware of splitted sessions, so they could list the client side session components. Another point to be supported is the detection of server-only components of some plugins, when the client with the "easy-to-grep" strings is missing. Particularly some actions like CANCEl or MOVE might require knowledge of service location.

Therefore each package for a specific session type has to support an appropriate function that checks the availability of this feature.

In general the following actions might support client side execution(not all of them might be implemented yet). Generally the localhost is the only client recognized by implicit split.

11.3 Command-Sets

11.4 Plugins

11.4.1 Main Dispatcher

11.4.2 Standard Plugins

11.4.3 Extensions

11.4.4 Subdispatcher

Common Concepts

The subdispatcher is one level below the top-level ACTION dispatcher and manages collaborative informations modules, which offer data to be used as a set in combination with data from other plugins. A typical application of this is the handling and tailoring of sorted data streams with various sort keys other than the originator only.

Therefore the task fullfilled by the subdispatcher is first a minor degree "horizontal" re-ordering of fields, which off-loads the maintainer from frequent adaptation of all plugins.

The second and major task is the re-ordering of the "vertical" datastream by sorting and filtering on an abstract data level. For example common conversion like the decision to uniquely present TCP addresses as IP addresses or DNS names is a typical overall content filtering with mediation to be applied.

The varios types of plugins partly share a common subset of data which could be used for intermixed tasks. This is for example the managment of dynamic data generated by activation of static data structures like VMs. This is conretely applied to VMs, where a LIST instance could be identified by a so called MAXKEY as a maximum sized common subset data-key in order to assure unambigious search results from the stored configuration data pool. Therefore the LIST and ENUMERATE actions provide both the keyword MAXKEY for output of an appropirate data-set to be used by the query tool ctys-vhost managing the internal VM database.

ENUMERATE

The following table lists the internal ENUMERATE input format from called plugins. This format is supported from each plugin by mediation of it's data from the specific data sources to a common internal canonical interface. One record is present for each interface of the VM, which is frequently more than one.

This dataformat is the common format not only for the internal subdispatcher, but also lterally for storage within the cacheDB and for internal data exechange, e.g. for pre-checks and validation of the CREATE action.

The data record is transfomed and presented in various formats and sub-sets as requested by the user. The final output is managed by the generic intermediate subdispatcher for ENUMERATE action.

Nr.	Field	Description	Common	Remap
1	ContainingMachine	Machine hosting a VM.	X	1
2	Label	User defined unique label.	X	3
3	ID	The path of the configuration file.	X	4
4	UUID	The UUID.	X	5
5	MAC	MAC address.	X	6
6	DISPLAY	Optional DISPLAY.		8
7	ClientAccessPort	Optional client access port.		9
8	ServerAccessPort	Optional server access port.		10
9	VNCbaseportVNCPORT	VNC base access port.		11
10	TCP	TCP/IP-Address.	X	7
11	SessionType	Type of session, a.k.a. plugin.	X	2
12	Guest-Dist	The distribution installed as guest.		12
13	Guest-Distrel	The release of the distribution.		13
14	Guest-OS	The guest OS.		14
15	Guest-OS-Rel	The release of the guest OS.		15
16	VersNo	The version of the VM config.		16
17	VM-SerialNo	An arbitrary serial number for VM.		17
18	Category	The category of the configuration.		18
19	VMSTATE	Configured state of VM.		19
20	HYPERREL	Release of the install hypervisor.		20
21	STACKCAP	The list of offered capabilites		21
22	STACKREQ	The list of capabilites required.		22
23	HWCAP	The offered capabilities.		23
24	HWREQ	The list of required capabilities.		24
25	EXECLOCATION	List of hostnames.		25
26	RELOCCAP	Relocation capabilities.		26
27	SSHPORT	Alternative port for SSH.		27
28	NETNAME	Network name of current interface.		28
29	RESERVED07	For future use.		29
30	RESERVED08	For future use.		30
31	RESERVED09	For future use.		31
32	RESERVED10	For future use.		32
33	IFNAME	The interface within the GuestOS.		33
34	CTYSRELEASE	The MAGIC-release-ID of ctys.		34
35	NETMASK	The netmask of current segment.		35
36	GATEWAY	The routing gateway.		36
37	RELAY	Local-Peer-Interconnection device.		37
38	ARCH	Virtual architecture.		38
39	PLATFORM	Virtual device.		39
40	VRAM	The assigned amount of RAM.		40
41	VCPU	The assigned number of V-CPUs.		41
42	CONTEXTSTRG	A private storage for the plugin		42
43	USERSTRING	A custom string from the user.		43

Table 11.1: ENUMERATE-Input-Format from Plugins

Nr.	Field	Description	Common	Remap
1	ContainingMachine	Machine hosting a VM.	X	1
2	SessionType	Type of session, a.k.a. plugin.	X	11
3	Label	User defined unique label.	X	2
4	ID	The configuration filepath.	X	3
5	UUID	The UUID.	X	4
6	MAC	MAC address.	X	5
7	TCP	TCP/IP-Address.	X	10
8	DISPLAY	Optional DISPLAY.		6
9	ClientAccessPort	Optional client access port.		7
10	ServerAccessPort	Optional server access port.		8
11	VNCbaseport	VNC baseport.		9
12	Guest-Dist	The guest distribution.		12
13	Guest-Distrel	The release of the distribution.		13
14	Guest-OS	The guest OS.		14
15	Guest-OS-Rel	The release of the guestOS.		15
16	VersNo	The version of the VM config.		18
17	VM-SerialNo	An arbitrary serial number.		17
18	Category	The category of the configuration.		18
19	VMSTATE	Configured state of VM.		19
20	HYPERREL	Release of install hypervisor.		20
21	STACKCAP	The offered capabilites		21
22	STACKREQ	The required capabilites.		22
23	HWCAP	The offered HW capabilities.		23
24	HWREQ	The required HW capabilities.		24
25	EXECLOCATION	Valid exec locations.		25
26	RELOCCAP	Relocation capabilities.		26
27	SSHPORT	Alternative port for SSH.		27
28	NETNAME	Network name of current interface.		28
29	RESERVED07	For future use.		29
30	RESERVED08	For future use.		30
31	RESERVED09	For future use.		31
32	RESERVED10	For future use.		32
33	IFNAME	The name of the interface.		33
34	CTYSRELEASE	The MAGIC-release-ID of ctys.		34
35	NETMASK	The netmask of current segment.		35
36	GATEWAY	The routing gateway.		36
37	RELAY	Local-Peer device.		37
38	ARCH	Virtual architecture.		38
39	PLATFORM	Virtual device.		39
40	VRAM	The amount of RAM.		40
41	VCPU	The number of V-CPUs.		41
42	CONTEXTSTRG	A private storage for the plugins.		42
43	USERSTRING	A custom string for the user.		43

Table 11.2: ENUMERATE-Output-Format of Sub-Dispatcher

INFO

LIST

The following table lists the internal LIST input format from called plugins. This format is supported from each plugin by mediation of it's data from the various data sources.

Nr.	Field	Description	Common	Remap
1	ContainingMachine	Machine hosting a VM.	X	1
2	Label	User defined unique label.	X	3
3	ID	The path of the configuration file.	X	4
4	UUID	The UUID.	X	5
5	MAC	MAC address.	X	6
6	DISPLAY	Optional DISPLAY.		8
7	ClientAccessPort	Optional client access port.		9
8	ServerAccessPort	Optional server access port.		10
9	PID	UNIX process ID.		11
10	UID	UNIX user ID(any format).		12
11	GID	UNIX major group ID(any format).		13
12	SessionType	Type of session, a.k.a. plugin.	X	2
13	C/S-Type	Client or Server flag.		14
14	TCP	TCP/IP-Address.	X	7
15	JOBID	JobID when available.		15

Table 11.3: LIST-Input-Format from Plugins

Nr.	Field	Description	Common	Remap
1	ContainingMachine	Machine hosting a VM.	X	1
2	SessionType	Type of session, a.k.a. plugin.	X	12
3	Label	User defined unique label.	X	2
4	ID	The path of the configuration file.	X	3
5	UUID	The UUID.	X	4
6	MAC	MAC address.	X	5
7	TCP	TCP/IP-Address.	X	14
8	DISPLAY	Optional DISPLAY.		6
9	ClientAccessPort	Optional client access port.		7
10	ServerAccessPort	Optional server access port.		8
11	PID	UNIX process ID.		9
12	UID	UNIX user ID(any format).		10
13	GID	UNIX major group ID(any format).		11
14	C/S-Type	Client or Server flag.		13
15	JOBID	JobID when available.		15

Table 11.4: LIST-Output-Format of Sub-Dispatcher

SHOW

11.5 Security Design

11.5.1 Accounts Impersonation

11.5.2 Access Location

11.5.3 System Resources

Chapter 12

CTYS-Nameservices

12.1 Runtime Components

The components of the nameservice are structured as depicted within the figure:9.3 on page:116 based on the ENUMERATE and LIST action.

The main utilities for queries and the for generation of cache database are as shown the ENUMERATE and the LIST action. In addition the tool ctys-vhost manages and pre-processes the raw data cached in the database. Therefore some pre-processed grouping and mapping is performed and a second-level cache database is generated from the first-level data which is a raw local storage. Anyhow, the raw data could already be used as it is, just some additional time consuming processing is performed for the transformation.

The ctys-vhost utility is the crucial facility of the UnifiedSessionsManager for the performant handling of data as well as the only viable network service for providing information to stacked VMs which are nested and therfore not neccessarily visible when the containing VM is offline. The internal data structure of ctys-vhost is as depicted in Figure:12.1 on page:156.

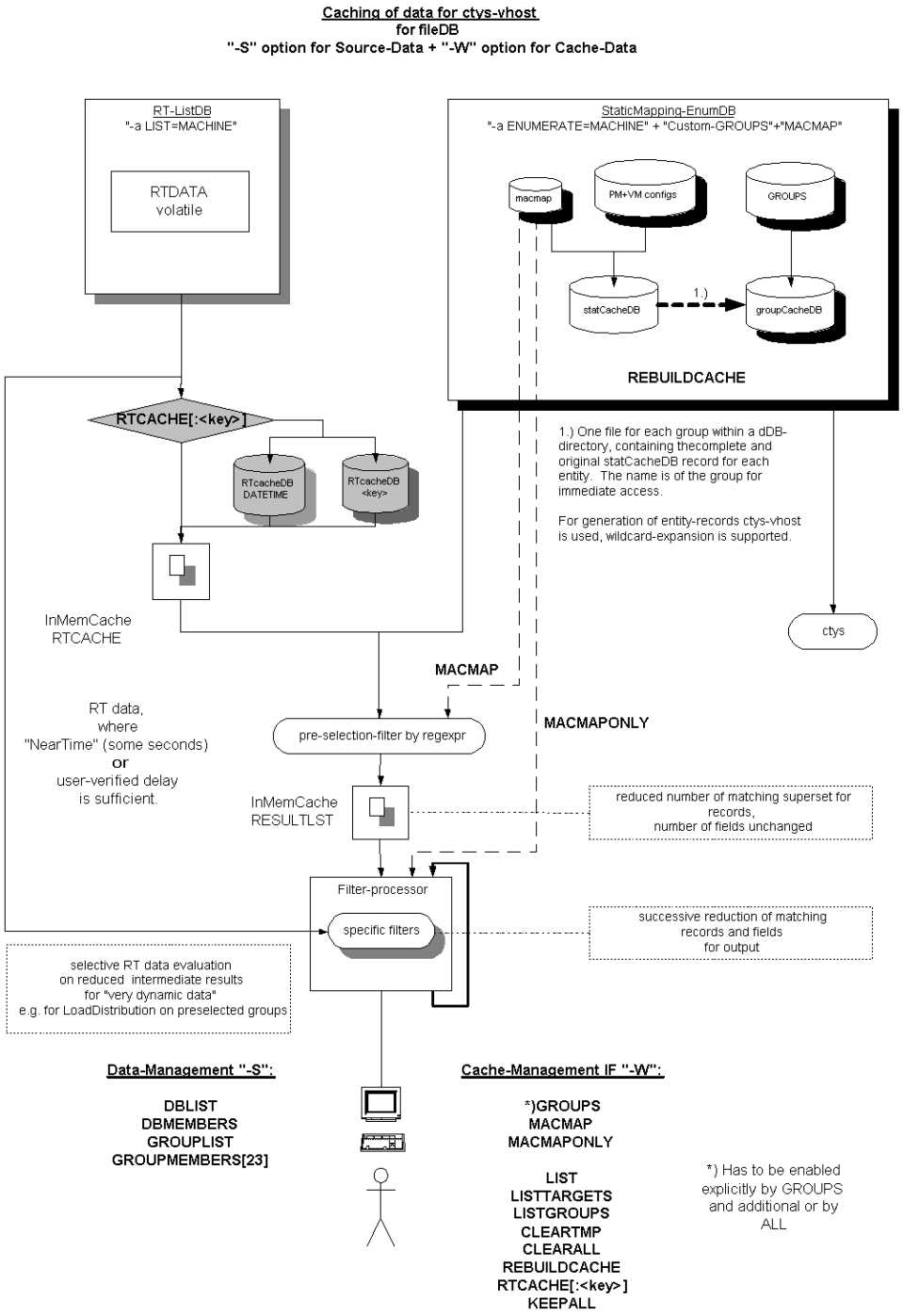

Figure 12.1: Nameservice components

Part III

User Interface

Chapter 13

Common Syntax and Semantics

13.1 General CLI processing

The common structure of the CLI call interface is defined by following basic elements.

```
<command> \
    <local-options> \
    [--] \
    <common-remote-options> <argument-list>


<argument-list>:=<argument>['('<context-options>')'][<argument-list>]
<argument>:=<command>
<context-options>:=<local-options>
```

Even though each part could contain a bulk of sub-elements, the basic structure is always the same.

The options are partly grouped and assembled by suboptions, which are scanned and operated by the involved plugins only.

Arguments, which are for "ctys" the execution-targets only, can contain their own scope of options and suboptions. These are pre-analysed on the caller site, but take mainly final effect on the execution site only. Therefore a common part for all remote sites and a specific set for each target could be defined.

Within the implementation of ctys the actual application of options within each scan is order dependant. The options are scanned from left-to-right, and in case of competition the last will win.

The scenario changes somewhat, when subjobs are generated. Each job is resolved with the global remote options and it's own options, finally superposed with the actual set pre-environment - including from the previous jobs. Due to some enhanced group resolution and the accessibility of several desktops, some reordering and grouping of tasks has to be performed. Thus the order of evaluation frequently changes for some of them.

Some of the options are prefetched for bootstrap phase itself until the CLI processing parts are active, this is e.g. the case for options related to dynamic and on-demand load of bash-libraries and plugins by "sourcing" them for initial bootstrap itself. Refer to the sources of ctys for extended inline documentation.

The masking of wildcards etc. could be done in addition to provided examples by any means of shell of course.

All keyword in parameters are converted and treated internally as uppercase. Though 'all' is equivalent to 'ALL', 'AlL', and 'aLL'.

For most of the keywords alternative short-cuts are supported, but due to avoidance of ambiguity caused by generic plugins, no wildcard expansion is foreseen. So short-cuts have to match literally. For additional information refer to specific options.

The additional extension as described in the following chapters are "group" instances for a set of hosts,

```
<argument>=(<host>|<group>)

<group>=<host>{1,n}
```

and macros, applicable as replacement-alias for any arbitrary CLI part and/or subpart, within any position except the <command> itself.

```
<command> <macro-alias>

    <macro-alias>=(
            [<local-options>]
```

```
    |[<common-remote-options>]
    |[<arguments>]
    |[<context-options>]
    |[--]
    |<any-resulting-sub-string-literal>
  ){1,n}
```

13.2 Options Scanners - Reserved Characters

The foreseen and implemented scanners are designed to allow implementation in a straight-forward manner by simply nesting loops and using sets generated from simple regular expressions. This is particularly important for simplification of user-implemented custom plugins. So following special characters are reserved for options definitions syntax:

'=': Seperator for option and it's suboptions. The reason for not using this as repetitive separator are "CALLOPTS" and "XOPTS", which are bypassed options for remote execution. These contain almost for sure a "=", but simplicity of the scanner is the priority here, so a second is choosen for repetition on groups.

',': Seperator for suboptions belonging to same group.

':': Seperator for suboption keys and it's arguments.

'%': Seperator for suboption argument values, will be replaced by space on final execution target "%==' '". Could be masked when required as literal by double-input "%%==%".

'()': Grouping character pair for target specific options belonging to a common target a.k.a. host.

'{}' Grouping arguments for multiple targets including their specific options belonging to a common high-level-target a.k.a. SUBTASK .

13.3 Hosts, Groups, VMStacks and Sub-Tasks

13.3.1 Common Concepts

The UnifiedSessionsManager supports for several areas of applications the bonding of multiple execution-targets to a combined group

entity. A group entity is a logical unit with it's own execution context, which could be founded by a specific set of attribute/option values and optionally implemented additional specific control of workflow.

Group objects are mapped within ctys to one or more specific subprocesses, which are called SUBTASKS. SUBTASKS could be allocated implicitly and/or by request, and are distributed locally and/or remotely, and could be used arbitrarily intermixed with the various SUBTASK types and or just in-process host execution.

The actual process structure and execution allocation is exclusively controlled by means of ctys, and partly presented as high-level attributes for call options. Thus flexibility and scalibility is inherent by the first basic design step, even though the "bash" is for now the only implementation environment and may not offer the full scope of available flexibility for process control and distribution.

REMARK:

Due to required preservation of the call-order for particular sub-tasks such as the type VMSTACK, the set of each call required to be preserved is heavily recommended not to be splitted across nested GROUPS with usage of "include" feature. This is due to partial re-ordering of the execution-targets when resolving include-trees, which could and frequently does lead to changed order of the target list during initial creation of internal batch list. This could not corrected during later execution.

The nesting of MACRO files is quite straight-forward and deterministic, thus could be applied once thorougly designed.

Curent version supports the following Sub-Tasks, which are described in detail within the following subchapters.

- SUBGROUP/SUBTASK
  Sets up a collection of <execution-target> as a flat set of entities starting within the same call-context, but executed independently.
- VMSTACK
  Sets up an execution context for a correlated set of <execution-target> as members of a hierarchical stack, thus sequentially

dependent on each other from-left-to-right. The mandatory part of the appropriate environment will be pre-set and forced to be kept. Semantical checks are automatically performed for generic dependencies.

The VMSTACK subtask supports the LOCAL and REMOTE mode, where the LOCAL mode is a centralized supervising controller for each remote entity. The REMOTE mode is a netsted self-propagating approach, where the controller just controls it's entity currently under control, and propagates itself to the next, when the current is enabled.

- VCIRCUIT
  Sets up an sequential relay chain with pre-assigned intermediate nodes for establishing an double-encrypted tunnel with various strategies. This particularly allows the deviation from the TCP/IP-routing, when appropriate intermediate nodes are available and accessible. Vulnaribility on intermediate nodes in case of compromized systems is reduced by secondary encryption of an SSH-IN-SSH([26, sshDefGuide]) tunnel.

  The bypassing of Firewalls for specific access groups is another typical application, where a strict control by means of SSH could be supplied.

  Anyhow, due to the pre-required access and specific knowledge of the actual network structure, this is less foreseen as a tool for intruders, but for those using some sophisticated network designs and equipment in order to fool them silently.

The subtask-entity could be used as a replacement for any position where an <execution-target> may be provided. A subtask could be customized with it's own context specific set of options, which will be - dependent on the specific type - permutated to all it's members. The basic systax is structured as follows.

```
<subtask>'{'<execution-target-list>'}'
    ['{'<subtask-arguments>'}']

<sub-task>:=(SUBTASK|SUBGROUP|VMSTACK|VCIRCUIT)
```

Figure 13.1: Subtask

13.3.2 Flat Execution-Groups by Include

The UnifiedSessionsManager supports for bulk access the concept of preconfigured groups. A group object, contains multiple instances of host objects and is a syntax element for replacement of an host entity, representing multiple nested instances. When providing one or more group entities, either intermixed with host entities or not, one main process(group) controls the whole set of subprocesses to be performed local or remote.

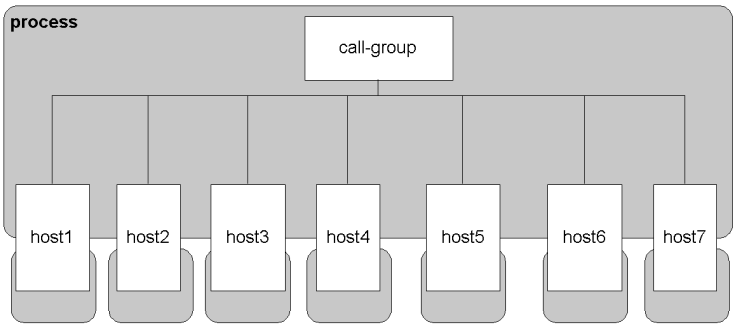

Figure 13.2: Groupresolution by Include only

This reduces a single combined multiple target execution of an (theoretically) arbitrary amount of instances to the call of a single group target. The group object can replace any valid execution target and supports the presence of context options within the definition file, as well as the additionaly assignement of context options to the group object itself at the caller's command line. These sets are combined by building a superset of both for each specific target. This is recursively true for the resolution of the whole tree of include dependency.

The definition of groups is simply defined by an ASC-II file with an easy to use syntax. The name of the group is simply the name of the file, which has to be stored within the search path:"MYGROUPPATH".

The literal name of the group file could be used as a fully functional replacement of any <execution-target>. Multiple group names are

supported as well as intermixed group names with host names.

Group objects has to be present on the callers machine, the tasks will be distributed to each member individually.

E.g. the following construct could be used for a group:

```
...myGroup1'( -g :A20)' myHost1'(-d 99)' myGroup'(-W -g :A10)'...
```

The group file can contain an arbitrary number of comma seperated host names on each line. Comment lines begin with a "#", empty lines are ignored. Any level of nested includes is supported, therefore the include statement has to be at the beginning of the line:

```
"#include <groupname>"
```

Circular inclusion will be detected at an default level of 20 and terminated than.

Due to integrated caching of expanded groups within the common nameservice, the "ctys-vhost" utility supports the visualization of configured groups in a structured manner, tree-view is planned to follow.

REMARK: The order of listed targets cannot be relied on, it is slightly varied, due to implementation ease by "includes-first" evaluation.

One example of groups expansion is given as follows:

```
ctys -a list MYGROUP01'(-d 99)' hostX MYGROUP02'(-d 3)'
```

```
group:MYGROUP01 "#include MYINCLUDE"
                "host01,host02"
                "host03"
```

group:MYGROUP02 "hostZ"

group:MYINCLUDE "hostA"
"hostB"

resulting call is - with MODIFIED ORDER:

```
ctys -a list                                          \
        hostA'(-d 99)' hostB'(-d 99)'                    \
        hostZ'(-d 99)'                                   \
        hostX'(-d 99)'                                   \
        host01'(-d 99)' host02'(-d 99)'host03'(-d 99)'  \
```

Any host entry within a group-file could be supported with specific context options. Thus the group feature implements an almost full scale MACRO feature for on-call access.

This could be helpful e.g. as predefined shutdown and suspend, and start-up groups. With GROUPS it is an easy task to:

- gracefully handle all PMs
- with the currently executed VM stacks,
- which are connected to one UPS,
- which signals an alarm.

Predefined escalation-shutdown groups, in order to expand the remaining backup-time for higher-priority tasks could be setup and preconfigured easily.

Generally could be said, nesting of braces is NOT supported, but chaining of braces IS. Permutation is performed for now only for the first level of group resolution, which might change, when implementing a some more sophisticated scanner/parser.

One specific point to be aware of is, that due to the following

1. re-ordering of entries
2. the "from-set-on" for all relevant, but not resetted values of the overwritten context options

it has to be underlined, that when using context options within a group file, all items has to be set explicitly, or none at all. Other wise a number of side effects might occur due to unexpected mixture and interference of options from various contexts.

This results technically from the decision to eas the design and implementation caused by support of limited namespace capabilities and the absence of "structured data assemblies" and associative arrays within the bash.

The tool "ctys-vhost" supports an interface with the option "-C" and "-S" for basic management of user defined groups and their pre-resolved caching due to performance enhancements.

13.3.3 Structured Execution-Groups by Sub-Tasks

The group feature is extended within the "ctys" script by the concept of subgroups, which is slightly different from include.

The inclusion of a nested group is performed once at the beginning of a call, and is resolved in a "hungry" style, by complete resolution of the whole dependency tree. See figure:13.2 on page:164.

The resolution of subgroups is performed by a delayed name resolution, which is executed as a seperate subprocess. The existence of an unresolved group within the defined dependency tree is checked immediately when matched for the existence of the non-included group definition file, this is done before starting the child process.

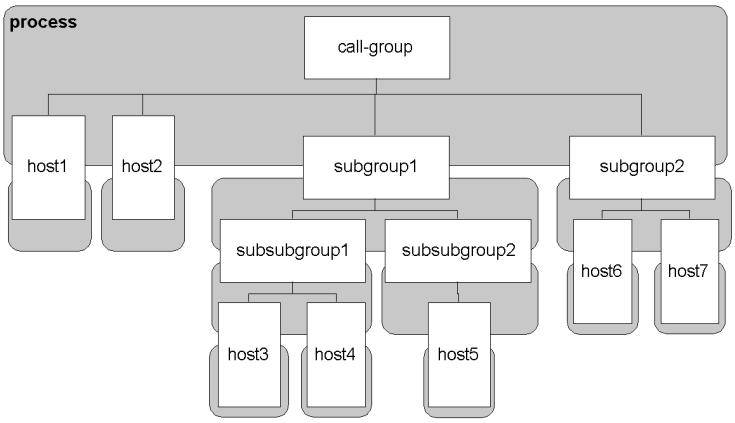

Figure 13.3: Groupresolution by Subgroups

When a valid definition file is present, the group is kept unresolved and treated within the job scheduler of the current process as an existing target for execution. Once the schedular performs it's accomplished internal batch queue, the delayed subgroup is executed locally within it's own copy, which resolves the first level of subgroups and executes them. It is important to mention, that the delayed resolution will be performed recursively, each layer in it's own process, thus this should be configured thoroughly.

The excution within it's own context could be implemented by various approaches such as own processes, and/or threads with various local and/or remote distribution philosophies, the advance of the actual used approach is it's simplicity and common availability for all known variants of the choosen bash environment.

The main advance of subgroups is the specific context of execution, where for example a completly different background operations mode could be established. One common example for this is the scanning of VM configuration files by "ctys-vdbgen" on machines with limited resources, where a sequential processing of multiple user accounts is required.

This case could particularly be of relevance, when scanning a Dom0 of a Xen based hypervisor.

The configuration of the amount of RAM for example is recommended frequently to be 256MByte for the most hypervisors, for the Dom0 of Xen frequently 512MByte are recommended. This is perfectly all right for an "idle-ing" machine. E.g. CentOS could be installed with 256MByte, but the install of OpenSUSE-10.2 should be performed with 1Gigabye, and may not work at all with 256MByte.

Another aspect secific for the UnifiedSessionsManager is the on-demand start of a hypervisor, which is e.g. a DomU in case of Xen. This is required for any user within the containing domain, in case of a first-level stack-entity the CREATE of a DomU has to be performed within the Dom0, with the appropriate access permissions of course. The same is true for ENUMERATE in case of scanning for stored VM configurations within the Dom0 in order to build a cacheDB. The performance could be degraded additionally by usage of a central NFS based VM pool for stack elements when a considerable amount of parallel scans are executed. When scanning multiple

stack-layers with multiple user accounts at once, which are hosted on the same machine, it could lead to dozens of scans at once on the same physical machine. Thus the this may influence the overall performance dramatically.

It has to be mentioned, that even though the current version of the UnifiedSesssiosnManager was designed and implemented with less priority on the required system resources, it works perfectly and with a good average performace with 512MByte of RAM, when one user onyl performes an ENUMERATE with an excessive file search and content scan. But requires 1GByte and more for multiple users within Dom0. When limited memory is available, the performance might degrade in case of multiple users dramatically, when continuous long-running jobs like ENUMERATE are executed.

The requirements are less when just CREATE actions by usage of local cacheDB are executed, which is the daily use-case for common usage. The overall performance depends in addition on the general load of the execution target of course.

Thus various approaches could be applied in order to reduce the overall execution time when multiple tasks are performed SYNCHRONOUS and in PARALLEL on different targets with varying present resources.

One possible solution is to reduce the number of tasks performed on the weak targets, whereas the better equipped targets might perform more parallel tasks. The solution within ctys is to use SUBGROUPS, which could be performed each with diferrent background-policies.

A more finegrained SUBGROUP hierarchy could be setup when required, allowing more variants of specific execution-call-contexts within one call.

It should be considered, that the client machine, requires for each SUBJOB an own fork of the ctys process, thus may have a sufficient amount of RAM, which is with some GiByte moderate for nowadays.

13.3.4 Stacks as Vertical-Subgroups

The VM-Stack implementation is by it's nature and thus it's design close to to a Sub-Group with just a fixed set of context options.

This is due to the inherent successive dependency of each upper stack-peer from it's "underneath" peer as a natural prerequisite for it's own execution.

Subgroups as a higher level design artifact are basically just a set of independent entities within their own but common execution context. The execution context could provide some basic level dependency such as "sequential execution", but is not defined to offer additional generic content dependency on framework level.

The neatless extension of the design concept of SUBGROUPs to VMSTACKs, extends the "flat-feature" and "horizontal-feature" by some abstract constraints based on the present generic execution control features. These concepts could be applied intermixed, but the nesting of VMSTACKS is not provided.

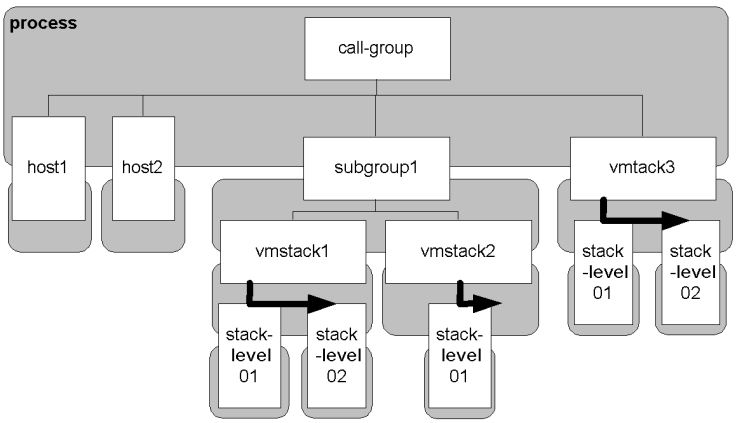

Figure 13.4: Combined Subgroups and Substacks

Due to the inherent execution depency by the "hierarchical vertical dependency" of the elements of a VM-Stack, the execution will be forced by the framework to set some generic attributes on framework level. This particularly controls the hierarchical execution dependency within the "vertical-feature" of a VM-Stack. In addition specific options for the activation of the stack-control are set.

- "-b SEQ,..."
  The VMSTACK feature(at least for now) requires the sequential execution of the parts of the requested stack. This is also

true, when the "REUSE" suboption of CREATE is set, because a dynamic pre-check for the availability of the next founding entitiy before execution of the hypervisor is mandatory.

The current version supports a "single-line" of a stack for a single call, thus "upper-trees" and "branches" of entities will be rejected. When multiple instances on a specific level are required, these have to be executed within multiple STACK-REQUESTS, which of course might share various parts of their lower branches. The STACK-REQUESTS could be combined to one "bulk-execution" call, and might not interfere erroneous, when all have the "REUSE" flag set. Anyhow, due to partly unavoidable polling, some repetition-counters and timeout values might be set appropriately, as they are for the most of the cases by the default values.

- "-b SYNC,..."
  The sequential execution implies the synchronous execution, because no parallel threads within a single VM-Stack call are supported.
- "-b STACK[:<max-stack-height>],..."
  The key STACK foreces the previous listed keys to be set as described and rejects any further changes. This key implies and forces to the processing of the whole set of following <execution-targets> as a member of one VM-Stack. Due to possible unintended calls with groups expanding to a mass of targets, a configurabble threshold value for the maximum of expected stack members is set by default CTYS_STACKHEIGHT_DEFAULT. This could be modified persitetntly and/or set call-by-call.
- -a CREATE= STACKCHECK :....
  The VMSTACK will be pre-checked concerning various aspects due to it's complexity once executed. These checks could be too restrictive for daily business and might not really be required, therefore some should be deactivated when appropriate.

  One specific candidate is the CONTEXT property, which represents the location context where the configuration file was orginally detected by ENUMERATE. This defines by default the "ContainingMachine" as a pre-requisite for the locality of execution. The configuration attribute EXECLOCATION controls this property, which is actually the PM/HOST attribute.

The default value is set appropriately for the several session types, and defines independently from the actual existence of additional requirements whether the machine is fixed to be executed on a specific location. A common reason could be caused e.g. by security, where a critical machine containing data and access keys for financial departments has to be fixed to a specific location only. The value ROADWARRIOR defines the VM to be executable anywhere, when additional pre-requisites are fulfilled. Other VMs, might be more restrictive due to their lack of support for stacking on other entities. The value LOCAL restricts the execution to the original scan location. Particularly emulator based VMs as QEMU, which in general could be executed anywhere, are set to the default ROADWARRIOR. This eases the initial creation of a cacheDB and requires a smaller amount only, due to the inherent flexibility of the initial execution location.

The complementary attribute RELOCCAP defines the change of a location for an active machine, as provided by means of the utilized hypervisor.

The STACKCHECK could be disabled partially or completely, what is foreseen for test cases primarily.

The Figure:13.4 depitcs by the symbolic arrows a probable execution sequence as a dependency caused by nested containment. Thus the "stack-level 02" are remotely executed within the instances "stack-level 01", as would be the "stack-level 03" within the "stack-level 02".

The previously mantioned basic checks for a VMSTACK include the consistency of the following characheristics of the stack.

- collectStackData
  Collects the data required for further analysis, thus performs the very first check for the availability of the requierd data.
- verifyCreateOnly
  This is a specific test for this version, where the combination of CREATE actions for VMs/PMs is supported only.
- verifyStacking

This checks the consitency in addresses of the actual call commands.

- verifyStackCapability
  This verifies the session type of the VM against the STACKCAP attribute, thus the availability of the appropriate hypervisor.

  Anyhow, due to the option of dynmically start different customized kernels for various modern OSs, the STACKCAP, which in case of a cached entity is a static snapshot only, might deviate from the last boot of the actual target. E.g. a VMware configured kernel instead of a Xen-ified kernel might currently be active, thus this check has some limitations concening the synchronity of it's decision base.

- verifyHardwareCapabilityStatic
  This verifies the compatibility of the hardware, as presented by the hypervisor to match the requirement of the GuestOS. This is particularly required for two properties, one is the architecture ARCH, which has to match the required CPU particularly for emulators such as qemu-ARM. The second is the virtual RAM, which might be exausted by the single VMSTACK call and/or by the actually running additional VMs competing for the available resources.

- verifyStackLocation
  This check verifies the location of the various stack entities. Therefore first the check of the bottom-level entity assures the location for the whole stack, whereas the additional checks verify the relative stack position of the upper layers, nested within the bottom element.

  The check covers several aspects to be considered for wider stacked operations, where the embedded entities are not actually aware, and if, cannot really be sure, where they are actually executed. The first aspect to be covered is the availability of specific resources at a specific physical location - namely machine - only. The location has to be verified for example in order to have access to a specific local hardware-peripheral, which might be available at a small number of machines - PMs - only. Also a specific driver of a VM, which probably is available on specific site only, sould be constraint. The second more generic, but possibly much more critical aspect is a possible

security flaw, when an intruder becomes able to fake a location in order to hijack the whole, or just a part of a stack. This becomes quickly clear, when an accounting machine, implemented as VM, contains probaly some specific data, or access keys. It has to be recognized, that the owner of the executing base machine is definitely the master of the nested upper part of the VMSTACK.

Thus at least a thoroughly performed pre-check for the actual locality before the execution has to be recommended.

The Figure:13.5 shows a 4-level stack example, which could be started with the following conceptual call example.

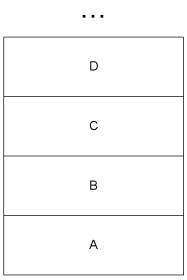

Figure 13.5: Stack Example for Basic Call-Interface

The inter-layer synchronity of the required sequential execution of the stack entities implies some specific constraints for the eventually choosen CONSOLEs. Thus the application of CONSOLE type of CLI has to be considered thorougly due to it's blocking character, which would block the whole upper stack, when applied. The application is still possible, but with the main intention of offering a means for application of the CMD feature. The non-blocking CONSOLE types will be silently forced into non-blocking and parallel operation by "-b async,par". The later independent creation of detachable CONSOLE types could be applied as usual. The usage of native HOSTs sessions is synchronous on session-level, conceptually seen as a non-layer stack-entity, which is embedded into a specific layer instead of being a layer entity by itself. Thus support for embedded execution of custom commands(CMD)[1]is assured by sequential left-to-right operation of a VMSTACK. ´

[1] For additional information refer to "X11-CMD" and "CLI-CMD" , application examples

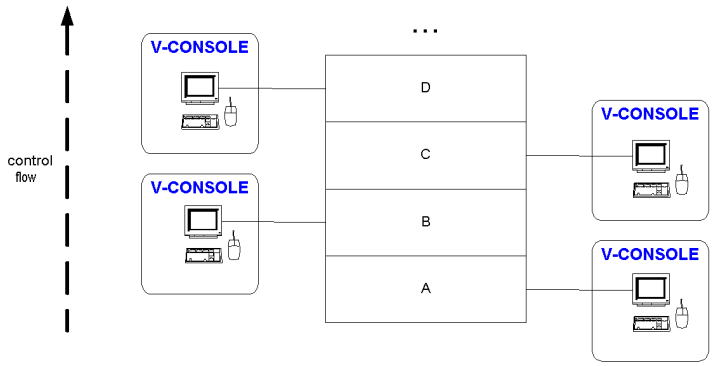

Figure 13.6: CONSOLE- and HOSTs-Asynchronity for Stacked-Execution

The stack-synchronity of the control flow for the operation, and though the application of the server components within sequentially dependent script-operations is assured by the "-b STACK" option, which is implicitly set. The attached CONSOLEs will be just "popped-up" as choosen.

The following call will incrementally startup the stack on the actual physical machine "A", after it's activation by usage of the relay wolExecRelayServer(refer to Section 23.1.4 'PM - Using Wake-On-Lan - WoL' on page 505).

```
ctys \
  VMSTACK'{ \
    wolExecRelayServer(-t PM -a create=l:A,WOL )\
    \
    A( -t SESSION-TYPE01 -a create=B ) \
    B( -t SESSION-TYPE02 -a create=C ) \
    C( -t SESSION-TYPE03 -a create=D ) \
  }'
```

This is controlled by the detection of the keyword "VMSTACK", which starts a pre-configured SUBGROUP with specific forced pre-

are available in Section 21 'HOSTs - Sessions' on page 429 .

assignment of the "-b" option as described before.

The same call splitted to two calls, a first for the WoL call to start "A" by usage of the "wolExecRelayServer".

```
ctys -t PM -a create=1:A,WOL wolExecRelayServer
```

A second call for incremental startup of the stack on the actual physical machine "A".

```
ctys \
  VMSTACK'{ \
    A( -t SESSION-TYPE01 -a create=B ) \
    B( -t SESSION-TYPE02 -a create=C ) \
    C( -t SESSION-TYPE03 -a create=D ) \
  }'
```

The current version just limits the allowed user suboptions for the "-b" option, but lets the remining to the responsibility of the user. This offers the flexibility for example to use intermixed hypervisors, authentication facilities, and CONSOLE types within different levels of a stack call. But some parameters could only be "late-checked" for applicability just before the final execution.

For this version no implicit creation of stack entities is supported, thus each CREATE has to be provided by the user, which could be combined to one call.

13.3.5 VCircuits as Sequentially-Chained-Subgroups

REMARK:

This feature is currently under development, and thus is possibly partly or at all not yet available. If so, it will follow soon within an intermediate post-release. Same is true for the full range of description.

The VCIRCUIT subgroup is by it's functionality close to the VMSTACK subgroup, which executes successivley commands of on a

set of logically vertical grouped host entities. The VCIRCUIT utilizes a chained set of machines in order to establish a temporary static encrypted tunnel. The peer-to-peer tunnel is in addition to its SSH based sections encrypted as a virtual circuit, providing a higher level end-to-end channel.

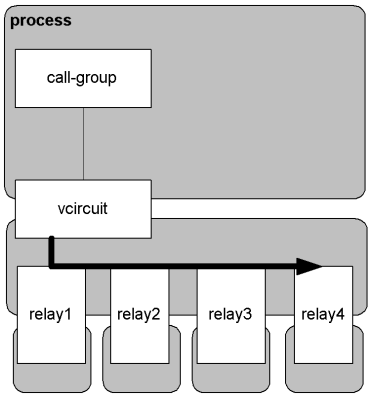

Figure 13.7: VCIRCUIT

13.4 CLI macros

The MACRO feature supports the usage of a predefined string alias as a literal replacement within any position of the CLI call.

Thus a macro can contain any part of a call except the command itself. The whole set of required options including the execution target or only a subset of options could be stored within a macro.

The macro and it's content are stored within a file which could be edited by each user or provided as a common defaults file. MACROs are resolved on each executing machine, thus even though a client could send a MACRO to the server, in current version the macro is resolved completely as the first step before the resulting call is processed and distributed.

A macro is defined within the default file named "default" which is searched in the order:

1. "$HOME/.ctys/macros/default"

2. "<actual-call-conf-path>/macros/default"

The <actual-call-conf-path> is evaluated from the resolved symbolic link of the call.

The following call syntax is provided:

```
MACRO:(
        <macro-def>
        |
        '{'<macro-def>'}'
      )

<macro-def> :=
   <macro-name>
   [%<macro-file-db>]
   [%OPTIONAL]
   [%(
       ECHO
       |EVAL]
     )
   ]
```

MACROs could be nested and chained as required. Even though the recursion depth could be arbitrary a counter is implemented, which sets a threshold limiting recursive processing. This is set by the configuration variable CTYS_MAXRECURSE. The variable protects all recursion depths, thus should be handled carefully. Default is 15 levels.

When macros are closely embedded into strings braces could be used, this could e.g. be applied in order to append contect options to predefined macros.

```
ctys '{macro:tst-subgroups-01}(-d 99999)'
```

Where the macro "tst-subgroups-01" is defined as:

```
tst-subgroups-01 = -a list SUBGROUP'{host1 host2}'
```

This expands a the end to:

```
ctys -a list host1'(-d 99999)' host2'(-d 99999)'
```

The keyword "MACRO" prefixes the actual macro alias with the following parts.

<macro-name>
The actual name of the alias to be replaced.

<macro-file-db>
The default macro file could be altered by this new file name. The "macros" directories will be scanned for a file with given name.

OPTIONAL
The given macro signed as optional, thus if it is not found it will be ignored silently. Else a missing macro leads to an error and abort.

ECHO
The given macro is inserted by "echo" command into the replacement position, which is the default behaviour.

EVAL
The macro is evaluated on the callers site by "eval" call and the result is inserted into the insertion position.

The MACRO feature could be combined with the GROUP feature in various ways, particularly the combination with the raw syntax of the supported SUBTASKs for SUBGROUP and VMSTACK is applicable.

The following example shows the call of a predefined SUBGROUP with activated remote debugging for the "permutated" targets resulting from the MACRO.

```
ctys '{macro:tst-02%test-subgroups}(-d 99999)'
```

The named MACRO-file-db "test-subgroups" contains here the test-case "tst-02" for a "LIST" action on two remote test-hosts, which is:

```
tst-02      = (-a list) SUBGROUP'{host01 host02}'
```

Thus the resulting actual call executed by ctys after MACRO and GROUP resolution is:

```
ctys '(-a list)' host01'(-d 99999)' host02'(-d 99999)'
```

This call suppresses for now the display of a header, just executes on the remote hosts and displays the actual data-rows. This is due to the missing assignment of a local ACTION, which is required as

an overall controller for actions displaying data in competition for the display.

The following call displays the LIST table including a local header.

```
ctys -a list host01'(-d 99999)' host02'(-d 99999)'
```

The following call in addition first collects data, thus does not poison the result data within the table with eventual ERROR messages and WARNINGs, but but them before the table.

```
ctys -a list -C raw host01'(-d 99999)' host02'(-d 99999)'
```

For additonal variations refer to the available generic options.

13.5 Shell Interworking

13.5.1 ctys as Sub-Call

13.5.2 ctys as Master-Call

13.5.3 Generic Remote Execution

13.5.4 Access to ctys Functions

13.6 Common Options

<callopts>
: Call options are passed literally to a remote command, therefore no intermediate processing is performed. White spaces are not supported and has to be replaced by '%'. E.g. "bash -e ls" is masked as "bash%-e%ls".

<xopts>
: X-options are passed similar to <callopts>, but to a X11 application. The user has to be aware of the single-hyphen and double-hyphen usage of the various X11, tools for their options. The core parts for geometry and title are set by ctys.

  Be aware, that some of X-options such as "-geometry" and "-name" are already implicitly utilized by other options, thus use this if, than CAREFULLY.

(SHELL|S):<shell>
: Replaces the standard definition of an interactive shell by

"CLI_SHELL_DEFAULT='bash -i'" for remote execution.

When setting an own shell the masking of SPACEs has to be applied by "%" in accordance to common ctys cli rules. The default will be applied on command line interface as:
"... -a CREATE=c:bash%-i ..."

The main difference to the CMD option is the execution of the given command without starting a new shell previously. Therefore available library functions of ctys could be called. For examples refer to Section 21.1.8 'Call ctys functions' on page 433 .

CMD:<cmd> Replaces the standard definition of a command execution shell by
"CLI_SHELL_CMD_DEFAULT='bash -c'" for remote execution.

Chapter 14

Core Data

14.1 Overview

The internal static configuration data handled by the "UnifiedSessionsManager" is based on the output records of the ENUMERATE action. This action is the founding part for generating the internal caching database to handle VMs and PMs and addressing the offline GuestOSs.

The ENUMERATE action scans local and remote filesystems and detects the configuration files of each active plugin. This is technically performed by calling an internal interface of each actual loaded plugin in the operational state ENABLED. This is performed on each enumerated execution-target and collected into a common database on the calling machine.

The data is stored in the MACHINE format, which is a semicolon seperated ASC-II record format, and could be imported to almost any database and spreadsheet. The description of the records could be displayed by usage of the common keyword TITLEIDX or TITLEIDXASC within each reporting action and tool.

The data scanned by ENUMERATE is pre-cached into a local database and managed by the tool "ctys-vhost" due to the processing time required for a filesystem scan. A similar reason is the included managemend of entities, which are potentially off-line when the query for specific attributes is performed. Thus caching supports reuiqred functionality for off-line PMs and VMs as well, as a reduction of the average query-time to about 0.6-1.0 seconds for databases with about 500-2000 entries(on a limited medium machine). In contrast to this, the actual scan of a deeply structure filesystem for a configruation file could vast minutes resulting from a simple attribute

value assertion. It could be said, that - depending from the filesystem - the average time gain is more than a factor of 100, as stated very conservative."

The second main application is the scan for actually operating entities by LIST, which is based on the internal representation of the common interface LIST, used for various queries and dynamic ID conversions. For example most of the LABELs are converted by usage of the LIST action. This could be cached too, but due to it's realtime or at least near-time requirement, onyl temporary short-time caches are utilized. Refer to "-C" ans "-b" options.

Even though the data is stored in the standard record format, some minor variations have to be applied to the various kinds of processing actions. For now basically three variations are distinguished:

ENUMERATE=<field-name><processing-options>
: The collected distributed static and raw data from the configuration files as provided by the user. Some minor add-ons, such as DNS and MAC resolution, are provided optionally.

  The main applications are the internal usage for dynamic path-extension of addressed targets in actions by usage of the UNIX "find" command, and secondary the pre-fetch of this information into a static cache database.

LIST=<field-name><processing-options>
: The dynamic data of all actual running plugins, this comprises not only the VMs and PMs, but also the HOSTs and TUNNELs.

  The LIST function is the working-horse for displaying and managing the actual dynamic state of all involved physical and virtual machines, including the contained operational facilities.

ctys-vhost -o <field-name><processing-options>
: The crucial interface to cached offline data for interactive user queries and the internal first-priority access base for configuration data queries. Refer also to ctys-vdbgen.

14.2 Standard Configuration Files

The plugins provided with the UnifiedSessionsManager could be generally subdivided into two categories by the way runtime data is

handled.

- transient runtime data
  These plugins handle dynamic data only, which is valid during their lifetime only. Thess are particlarly all HOSTs plugins, such as CLI and X11, where particularly temporary system IDs with temporary LABELs are utilized as aliases.

  This category of data is availabe for LIST action only, and cannot be enumerated. Anyhow, the dynamic instances of the persisten category are included in LIST action too.

- persistend runtime data
  These are mainly VMs, but PMs also, where the majority of required system data is defined within persistently stored configuration files and within the required runtime images.

  This category of data as stored instance attributes is available by usage of ENUMERATE and could be therefore prefetched and cached. The entries have to be defined and maintained by the user as supported for the different plugins.

Due to the integration of various hypervisors with different originators, the configuration data differs naturally more than having equal parts. The integration into one more or less neatless and at least basically unified interface is one of the main goals of the UnifiedSesssionsManager. The limiting edge of forcing compatibility is reached, when the processing of the configuration data for the varios hypervisors has to be handled. Therefore the following file-extensions with additional ctys-fields and records are supported. These are the file-extensions, which the plugins specific ENUMERATE actions are aware off, and though could be processed by ctys. Other file extensions will be ignored, and therefore are not accesible.

Plugin	File-Extensions
PM	conf
QEMU [1]	conf, ctys
VMW	vmx
XEN [2]	conf

Table 14.1: Supported File-Extensions

Basically two types of files could be distinguished, the configuration files with pure configuration data (conf, vmx), and mixed files(ctys), containing configuration data and/or executable script code, which is defined to be bash-code.

Particularly for QEMU, due to the original command line interface only, some wrappers are applied for various reasons. Therefore, beneath the VDE/VirtualSquare wrapper for management of the network interfaces, the ctys-wrapper is introduced in order to handle the flexibility of the call interface and the amount of call options offered by QEMU.

14.3 Common Data Fields

This section represents the core set of data which is used in several actions. Some of it's members are varied within specific call contexts when applied, and are therefore in addition specialized within the following sections.

ARCH
: Virtual architecture presented to the GuetsOS and hypervisors of upper-perr-stack-layer.

BASEPATH|BASE|B
: The path-prefix for the search root by UNIX command "find" to fetch all present configuration files within the subtree. This could be a list of nodes to be scanned, as depicted in the definiton of <machine-address> .

  This attribute identifies search groups of VMs as stored and organized within parts of the filesystem. Thus defines scopes of entities to be visible only for additional selection criteria.

  One typical application is to define a set of VMs within a directory with access permissions for a specific group of users only. Therefore canning and caching of data from that subtree is required for the permitted group only. Due to the supported input parameter BASEPATH of CREATE action, this could be used for views of work-scope organization, as well as basis

[1] For QEMU some extended control of BOOTMODE could be applied in combination with PXE boot.

[2] Some specific variations to name-conventions are applied on XEN in current version.

for load-balancing and various task dispatching groups. Several views could be organized by usage of symbolic links.

This parameter is applicable to VMs and PMs only, not to HOSTs.

CATEGORY|CAT
: The category of the plugin, which could be for now one of: HOSTs, PMs VMs.

CONTEXTSTRING|CSTRG
: A private context storage for the plugin.

CTYSRELEASE
: The so called MAGICID describing the current release of the UnifiedSessionsManager which created this record. Therefore each record could be traced to it's originator for debugging and compatibility reasons. This is somewhat handy, due to the actually distributed creation of the semantics at least, which is performed as a standalone task on each executing target by usage of the local hypervisor and ctys.

This is forseen to handle varying data sources of course.

DIST
: The distribution installed within VMs guest or PMs. This parameter is applicable to VMs and PMs only, not to HOSTs.

DISTREL
: The release of the distribution.

EXECLOCATION
: Defines the possible execution locations by a customizable list of possible execution locations. Thus various distribution policies could be implemented including specific views of an upper-layer semi or fully-automated algorithm.

The availability of the appropriate hypervisor has to be considered by the editor, else a missing type will be detected by an error when execution starts. This could change also dynamically, e.g. when during boot time different kernels providing differen hypervisors are choosen.

Due to distribution algorithms which rely on this set when configured ans activated by the RELOCCAP option, the value

of this parameter should be maintained thoroughly.

The following key values are supported for EXECLOCATION:

- LOCAL
  Could be executed at the install location.
- ROADWARRIOR
  A VM which could be started at arbitrary location. Anyhow, the availability of the specific hypervisor is still required.

GATEWAY
: The internet Gateway.

HWCAP
: The offered virtual HW capacity by the VM. This is particularly forseen to setup specific devices, which are physically colocated to a specific PM and are accessible local only. Examples might be specific HW-Test-Devices, as well as Machines. Another example are DVD-Recorde, Tape-Drives, specific security devices, or the required DMZ, in order to limit risks by openning connection with piercing of firewalls.

HWREQ
: This parameter is similar to the HWCAP parameter, but describes the required HW.

HYPERREL|HYREL HYPERREL|HYREL
: Release of the hypervisor used for installing the VM.

ID|I
: The ID of a plugin type. The syntactical data type varies for the miscellaneous plugins. For configuration-file based plugins, this is the filepath of a valid configuration file, unique on the executed machine. The plugin types of PMs and VMs generally support a configuration file.

  For dynamic plugins with temporary and volatile IDs, like CLI, X11, and VNC, the identifier represents an arbitrary numerical identifier, which is returned by the hosting system and/or the executed software component.

  Following current exceptions and specifics apply:

XEN
: The value is the configuration path statically unique on local host, common to IDs of other VMs.

  The volatile domain-ID is handled - due to hypervisor architecture and structural and dynamic means of accessibility - similar to an ordinary "UNIX-pid".

HOSTs
: For plugins of type HOST, which are more or less simple processes offering specific services, the "UNIX-ID" is utilized.

  The "UNIX-ID" could consist of several kinds of entries. A common example is VNC, where the entries semantic could be one of:

  - DISPLAY = VNC-port-offset
  - DISPLAY = VNC-port

  - Any of above could be context-specific, and utilized more or less correlated by any other FBP-aware application too. E.g. vncviewer for XEN, QEMU and WMWare-Workstation 6.

  For the CLI plugin the "initial-call-pid" of the topmost UNIX process is used as ID.

  So, it is just an abstract ID, no generic overall-algorithm for it's calculation is available. The only requirement is uniqueness within the required execution scope, which additionally could be deactivated by the "-A" option.

IFNAME
: The name of the interface within the GuestOS, which is correlated to this data record.

  This may vary due to several reasons, thus the synchronity is within the responsibility of the user.

JOBID
: The internal Job-ID assigned by ctys. The Job-ID is relevant for CREATE action only, though the remaining are just temporarily active, could be called "transient" actions. The CREATE action itself is transient too, but it's entity might be a

"longer running" item, thus will be called "qualified as persistent" here, emphasizing it's existence after the final return of the CREATE request.

Currently not all types of plugins assign persistent JOBIDs, thus could be listed only during initial execution. One example is the PM plugin when used with WoL.

The data required for display of JOBID is stored within a temporary file related to the PID of the item as displayed by LIST action. This data could be stored in shared mode, which enables anyone to display the full size of the LIST records, or it could be stored as private data, which just grant access permissions to the owner. This is controlled by the variable MYTMPSHARED within the configuration file "ctys.conf" and/or by pre-setting environment variable "MYTMPSHARED=NONE".

The SHARED usage might not be security relevant, due to usage of the private data first with priority, whereas "not-own" entities from shared directory are controlled by systems security facilities.

LABEL|L
: LABEL is a user defined alias as a user-friendly replacement for the ID. The various plugins set different requirements for the LABEL. So the Xen plugin requires a mandatory domain name for a DomU, which is used as LABEL. The VMware plugin utilizes the optional "displayName" as LABEL. For the QEMU an own configuration file format is defined. The PM plugin uses the DNS name as LABEL.

  For the HOSTs plugins the label is a call parameter, which is associated to the callee, temporarily valid for the current session only.

  Any contained ":"-colon will be replaced by an "_"-underscore.

MAC|M
: The preconfigured Ethernet MAC address. In current release only static configured MAC addresses are supported. Particularly any mapping information of associating Ethernet addresses with TCP/IP addresses has to be statically assigned,

and could be generated from DHCP configuration files and/or ping+ARP caches. Address pools of DHCP are not supported.

Only applicable to VMs and PMs.

REMARK: In case of multiple interfaces for an instance, each interface is enumerated as a seperate entity and eventually stored in the cache database.

NETMASK
Internet Netmask.

NETNAME
Name of the interface as to be used for external access. This is frequently the DNS name, which is recommended to be configured within DHCP. This should be done, even though the interface is probably used "addressless", e.g. for sniffing purposes. The inclusion within teh DHCP database provides automatica conversion into cacheDB and thus enables the usage within the UnifiedSessiosnManager. This is required for utilising WoL, where the actual interface on the host might be addressless(refer to Section 23.4 'PM - Using Wake-On-Lan - WoL' on page 510).

OS|O
The OS running witin the queried PM or VM. Not applicable to HOSTs category.

OSREL
The release of the OS.

PM|HOST
The TCP/IP address of the hosting machine, which is derived fom the "uname" output. This field exists for all local interfaces and has to be distinguished from the NETNAME.

This could be a PM, which is the founding physical machine, running the whole VM stack contained, or in case of an contained entity within the VM stack it is a VM itself, executing virtually as a PM for a nested upper stack VM itself.

PNAME|P
Almost the same as <ID|I>. This is due to the usage of filepathname of the configuration as an unique ID at least within the namespace of a single hosts filesystem.

RELAY
: The interface, bridge, switch/hub, or the router, which interconnects the VM to the network. Could be a host-only and/or an external connection.

RELOCCAP
: The capabilities offered by the VM for relocation of it's execution base. The current version supports the following values only, which could be applied in combination:

  - FIXED
    The VM could not be relocated at all. It is executable at the install location only.
  - PINNED
    The VM could not be relocated once it is started. A common reason could be the attachement to a specific hardware device, which possibly might be even available locally only.

    An example may be a debugging device for an embedded system, which is accessable by LPT device on local PM only.
  - ROADWARRIOR
    A VM which could be allocated and reallocated arbitraily, as though not specific requirements to the execution base is given. Anyhow, the availability of the specific hypervisor is still required.
  - <EXECLOCATION>
    The VM could be executed on a member given by the EXECLOCATION parameter only, which itself could contain GROUPs and MACROS. The execution string will be assembled by evaluating any extended-distribution criteria on the VM.

    These include the load in the sence of internal load to a VM limited by additional facilites, not just the pure processing capacity of the physical CPU. In addition some restricted and/or limited resources have to be assigned, which could be specific devices. E.g. a special type of printer, plotter, or cutting machine, which has limited access due to serialization via a batch-queue. This requires the distribution of the VM to another stack located on a differen physical machine, even though the actual load on the first

targeted machine might suffice a simple avarage-CPU-load balancing criteria.

SERNO
: An arbitrary serial number for the VM stored in the configuration file. This number should be unambiguous.

SPORT
: Server access port for execution of an administrative TCP/IP connect, seperated from the user access. This is the raw port to be used for server specific admin tools, which is different from user's client access.

  For XEN this port is not supported due to security reasons.

  For QEMU this port represents the monitoring port as a UNIX-Domain socket with specific naming convention.

SSHPORT
: Alternative port for "-p" option of SSH, default when absent is given by the system as "22". Multiple ports in varying sets for each interface are supported by OpenSSH. For information refer to *"OpenSSH"*[119, OPENSSH] and *"SSH The Secure Shell"*[26, SSHDefGuide].

STACKCAP|SCAP
: The list of capabilites of the embedded support for the upper-peer-stack-level.

STACKREQ|SREQ
: The list of capabilites of the required support from the founding bottom-peer-stack-level.

TCP|T
: The ip addresses of the GuestOS interfaces running within the VM. Each interface could have multiple assigned IP addresses.

  When enumerating the IP addresses, the MAC entries and the IP entries are scanned and correlated to each other based on the given numbers or the order only. The usage of the ordering position as index is applied specifically to Xen, due to lacking a numbering scheme for it's interfaces.

REMARK:

In case of multiple interfaces and/or addresses for each address of - a so called "multi-homed" machine - a sepereate entry is generated, thus it is listed as a seperate host entry. An interface without a MAC address is currently accepted, but generates a warning.

TYPE

Output of the type of session, either of category VM, PM, or a HOST by it's plugin name. The type of a session is to be used for the "-t" and "-T" options.In current version the following sessions are supported in the base set: CLI,X11,VNX,QEMU,VMW,XEN,PM

USERSTRING|USTRG

A string to be customized by the user, forseen as as a reminder to be displayed only.

UUID|U

The UUID is not neccessarily required, even though providing a quite well fitting globally unique identifier. The value could be generated e.g. by "uuidgen", but should be used as provided in case of PMs for hardware devices. For VMs and GuestOs it could be generated by the tool "ctys-genmconf".

The uuid is generally applicable for VMs and PMs only.

OSREL

The version number of the installed GuestOs distribution within the VM or PM.

The uuid is generally applicable for VMs and PMs only.

PLATFORM|PFORM

Virtual device, which is a unique identifier for the virtual hardware, either a PC-base, Server, or an embedded device.

VCPU

The pre-configured number of V-CPUs.

VERSION

The version of the VM config.

VMSTATE|VSTAT

The following values are applicable as actually stored attributes. The values could used in query tools either by their literal values, or by choosing a processing-only meta attribute for the

selection of a sub or superset.

The state value is semantically checked when generating a cacheDB by "ctys-vdbgen" and "ENUMERATE" . The post-processing and analysis tool "ctys-vhost" for now just does a generic pattern match on the record-stream from the cacheDB. Ambiguity has to be avoided by the user.

Additional values could be defined by the user and will be addeded to the cacheDB. Due to some semantic checks in order to detect mistyped standard attributes, these are required to be deactivated by the CUSTOM key before the position of the arbitrary key. The CUSTOM key is valid for any following key not matching a pre-defined.

REMARK:
Attribute values have to be stored literally as uppercase, the later match by scanning via ENUMERATE is performed as uppercase only.

The following values are predefined standard states:

ACTIVE
The VM is actively participating in operations, thus ready to be used in a production environment.

BACKUP
The VM is a backup of an existing VM, not neccessarily, but recommended of ACTIVE state.

TEMPLATE
The VM is a template to be used as custom base for productive VMs. The VM itself could be operable, but does not require so.

TESTDUMMY
The VM is a installed configuration only for testing and validating the basic functionality of the VM.

The processing attribute MATCHVSTAT provides means and additional operations-attributes for selection of subsets.

VRAM
The pre-configured amount of RAM.

VNCBASE
Base port for calculations of ports from display and vice versa. The default is 5900.

VNCDISPLAY|DISP
: DISPLAY to be used by XClients, which in case of VNC is already calculated by usage of context-specific PortOffset.

VNCPORT|CPORT
: Client access port for execution of a TCP/IP connect. This is the raw port to be used for vncviewer or proprietary clients with their own MuxDemux-dispatcher. This is required for example with VMW when using the workstation product of version 6.

14.4 Common Processing Options

CTYSADDRESS|CTYS
: A fully qualified address to be used within ctys. This includes the complete address for the whole execution-stack of the destination instance, beginning with hosting PM.

  Whereas almost any other output is just a subset of the generated static database, this value is the result of the assembly of multiple items to a complete address for an unambiguous execution path. The namespace could be the private network or even the global network, when globally unique PM addresses as FQDN are used.

  A typical addressing of an entity within a stack is:

  ```
  <host>[<vm-level-1>][<vm-level-2>]<host-acces>
  ```

  For a concrete example this results to:

  ```
  lab00.tstnet0.com[l:tstDomU01][t:tst3]'(-a create=l:myVNC01)'
  ```

  or by an aternative address pattern:

  ```
  lab00.tstnet0.com\
    [l:tstDomU01]\
    [p:/home1/qemu/tst3/tst3.ctys]'(-a create=l:myVNC01)'
  ```

  The syntax describes a VM stack path to be used in order to execute an action within the topmost element.
  Intermediate stack entries will be created by usage of default values when missing, default values are assumed for any missing option. Therefore the provided address information is accomplished from the required cached data by usage of ctys-vhost.

The execution only performs, when the reuquired data could be queried unambiguously from the available database. Else additional key information has to be provided by the caller.

The following example shows the usage of different authentication methods to the GuestOs.

```
lab00.tstnet0.com'(-Z NOKSU,SUDO -z pty,pty)'\
  [l:tstDomU01]'()'\
  [p:/home1/qemu/tst3/tst3.ctys]'(-a create=l:myVNC01)'
```

Even though the machine "tst3" could be directly addressed when in bridged mode and is accessible, namely already running, in case of CREATE and CANCEL the access to the hosting system is crucial. In case of CREATE, the extended addrssing schema contains the physical loaction where the stack entitity has to be executed, which could be extended by usage of wildcards for load-balancing or specific service distribution. In case of CANCEL the hypervisor could be involved into the shutdown, which is hidden else for access from within the GuestOS.

DNS
: Output of TCP/IP address (any valid for the VM). This option supports the name representation as reported by DNS, for the numerical representation refer to IP.

ATTENTION: Only the first match will be listed when multiple addresses are present for the same entity.

IP
: Output of TCP/IP address. This option supports the numerical representation, for the DNS name representation refer to DNS.

MACHINE
: Complete records matching the <regexpr-list> in terse format for postprocessing. The output is a semicolon seperated list of record, compatible with most spreadsheet applications . A title with the actual canonical field indexes could be displayed when combined with TITLEIDX . The extended variant TITLEIDXASC displays additionally the common column indexes for spreadsheet forms supporting the manual modification.

MATCHVSTAT
: The MATCHVSTAT key supports selective operations on stored VM configuration records. Therefore in addition to the stored values of VMSTATE some operational attributes are defined, controlling sub and supersets.

  The following values are applicable for attributes controlling the match process only and are therefore not stored literally.

  ALL
  : This simply sets the value to be ignored and matches any present and valid entry. When applying to ENUMERATE on stored configuration files, the MAGICID still will be applied and might superposition the semantics of the VMSTATE attribute due it's higher severity.

  CUSTOM
  : This key deactivates the validation of keywords, which allows arbitrary keywords to be used. This opens particularly the usage of custom VMSTATEs for definition of various scopes of sets to be included into the cacheDB.

  EMPTY
  : This value simply sets the select to non-existing state.

  PRESENT
  : This value simply sets the select to any but present state.

  The default match value is:"ACTIVE|EMPTY".

MAXKEY
: The maximum common set of attributes for LIST and ENUMERATE.

PKG
: The list of packages, a.k.a. plugins, to be displayed, any other will be suppressed. The syntax is as common:"PKG:pkg01%pkg02%...".

SORT[:<sort-args>]
: Sorts the body of table with given scope on the column of defined <sort-key>. This is mainly a sort on the first column. It first collects therefore the whole data of each machine, before displaying the result almost at once. No progress indicator is shown. Due to the smaller sort-scope the partial delays might not be too long.

Additionally the option "-C" influences the scope of sort, where without activated caching the scope is each executing machine, leading to a concatination of sorted sub-lists. When caching of the complete and raw result is choosen, the scope of sort is the whole result, displaying the list with sort applied to the complete set of records. Anyhow, when the first file is the PM/VM the result should be the same by default, until a specific sort-field is selected, which deviates from the default-field=0/1.

`<sort-args>=[ALL|EACH][%UNIQUE][%<sort-key>]`

ALL
: The sort is performed on top-level spanning the whole resulting table content.

EACH
: The sort is performed on level spanning solely each of the executing machines. The result is therefore grouped by execution targets.

UNIQUE
: Activates a pre-final filter for call of "sort -u".

<sort-key>
: Defines a sort key as "-k" option for "sort -k <sort-key>". The <sort-key> is the column index of the resulting outout table as displayed, enumeration is an increment beginning with "1".

TAB_GEN:<tab-args>
: Defines tables for formatted output. A simple set of macros is defined for the setup of a table definition compatible with most spreadsheet applications.

TERSE
: Lists the displayed items in machine processable way, it is the same format as with "-X" option.

TITLE
: Optional "title" could be applied for header listing of field/column names.

TITLEIDX
: Almost the same as "title", shows in addition the absolute and canonical field/column positions for addressing when a generic table is defined. The resulting output format is defined as the

list of all actual selected field names, each displayed with it's canonical index. For the output of all fields TITLEIDX has to be combined with MACHINE .

FIELD(index)

The current implementation for ENUMERATE yields to the default:

ContainingMachine(1);ID(4)

The complete record is displayed in combination with MACHINE as:

ContainingMachine(1);SessionType(2);Label(3);ID(4);UUID(5); MAC(6);TCP(7);DISPLAY(8);ClientAccessPort(9);VNCbasePort(10); Distro(11);OS(12);VersNo(13); SerialNo(14);Category(15)

The output is compatible with various spreadsheet applications .

TITLEIDXASC

Almost the same as TITLEIDX, but with additional display of column indexes for various spreadsheet calculation programs.

USER:<user>%[<credentials>]

The user to be used for native access to the <action-target>. This is required frequentyl for ACTIONs supporting the "Peer" mode and the "Auto-Stack" mode . Default useris the same as used for authentication on the execution target. The default authentication method is determined by the login target.

The alternative user could be provided either without specific credentials for usage with a configured network based authentication, or with one of the supported types of credentials. Following keywords are case-insensitive.

```
USER:<user-name>[%<credentials>]
```

The following applies to the <credentials>, which could be of various types.

```
credentials:=<credential-type>%%<credential>
```

Current version supports pre-configured network authentication only, either interactive, or by a specific protocol like GSS-API/TLS, thus the <credentials> field is not yet supported.

14.5 Specific Variations

BASEPATH|BASE|B:<output>

```
(BASEPATH|BASE)[:[<top-level>][%<bottom-level>]]{1,n}
```

The path-prefix for the configuration file of the current VM. This identifies search groups of VMs as stored and organized within parts of the filesystem.

Due to the supported input parameter BASEPATH of CREATE action, this could be used for views of work-scope organization, as well as basis for load-balancing and various task dispatching groups. Several views could be organized by usage of symbolic links.

GROUP
: The actual group id of remote server process.

USER
: The actual user id of remote server process.

PID
: The pid of remote server process.

TUNNEL|SERVER|CLIENT|BOTH
: List all selected types of connections on selected host. Default is S, which is similar to the definition of a session.

  The following sets could be selected:

  TUNNEL: tunnels only

  CLIENT: clients only

  SERVER: servers only

  BOTH: Yes, ehhh, all three.

14.6 Generic Tables

Several actions, particularly the GENERIC class of calls LIST, ENUMERATE, SHOW, INFO support data to be displayed in multiple specific views. The same applies for some support tools, particularly "ctys-vhost". The views may vary form task to task and should emphasize different topics.

Therefore the output could be adapted by the user with generic tables, which support a simple syntax with required minor knowledge only. These custom calls, which are based on a suboption for the specific action, could be stored as a MACRO and reused later. The recursive MACRO resolution supports for modularized table definitions which could be reused within the same ACTION, but due to canonical standard parts of some ACTIONS as LIST and ENUMERATE, also partly within multiple ACTIONS. An example could be found in Section 25.3.3 'Combined MACROS and Tables' on page 531 .

REMARK: Currently the actions LIST , ENUMERATE and ctys-vhost , support generic tables only, others will follow within next versions.

The common syntax for definition of a generic table is the following snippet of syntax, which has to be a supported suboption of the called ACTION.

```
-a <action>=(
              <action-other-subotps>
              [TAB_GEN[:<tab-args>]]
            )

    <tab-args>=<idx>_<colname>_<width>[_L][%%<tab_args>]{0,n}
        [,titleidx]
```

Each field entry has to be seperated by a double percent character "%%", as this is a parameter for the table processor itself, not the cli.

TAB_GEN
: The generic table processor to be invoked for evaluation of table parameters.

<idx>
: The canonical field index as provided by the ACTION. For the display of the actual values refer to "TITLEIDX" .

<colname>
: The name of the column to be displayed in the table header.

<width>
: The width of the table, which will cut the entry to the given value, if the size is exceeded.

B
: This optional key switches to clipping and insertion of a break. The table is expanded in it's length for each of required breaks. The cut is made arbitrarily, without recognition of the actual semantics within the specific field.

REMARK:
: In current version "B" is not compatible with the SORT option.

L
: This optional key switches to leftmost cutting of fields, clipping it to it's trailing part, when the <width> is exceeded.

title,titleidx,machine
: Any ACTION has to support a mandatory suboption TITLEIDX in addition to the implementation of TAB_GEN.

The "TITLEIDX" option displays the titles of each field with its positional index parameter as supported by the canonical record for MACHINE suboption. This is the index to be used by the underlying generic awk-script for the positions to be printed.

The <colname> parameter is case sensitive and therefore displayed literally. Restrictions for the available character set are the exclusion of reserved ctys-characters and the exclusion of any WHITESPACE, including CR. Comments and Whitespaces within the macro file are ignored.

Chapter 15

Address Syntax

15.1 Basic Elements

The addressing facility and therefore the namebinding is splitted into a logical description as a general view and it's conrete adaptions which could be implemented by multiple representations. Whenever it is possible any implementation is kept as close by it's syntax to the logical description as possible. The first version [144, ctys010303a01], and now this second public version uses a meta-syntax as might be intuitively understood.

The foreseen and implemented syntax scanners are designed to allow implementation in a straight-forward manner. Therefore some characters are reserved allowing an simple implementation of hierarchical structured syntax definitions by nested loops.

The full set and description of all reserved characters is given in the chapter "Options Scanners - Reserved Characters".

'=' Seperator for option and it's suboptions.

'.' Seperator for suboptions belonging to same group.

':' Seperator for suboption keys and it's arguments.

'%' Seperator for suboption argument values.

'()' Grouping character pair for target specific context-options belonging to a common target a.k.a. host.

'{}' Grouping arguments for multiple targets including their specific options belonging to a common high-level-target a.k.a. SUBTASK .

Anyhow, once the first version covers at least the full intended first-level scope of functionality and could be released as stable, an ASN.1 version with the full standard range of syntax documentation will follow. This is intended to be accompanying the first LDAP based release.

Therefore the current syntax description may not yet formally be absolutely correct, but may definitely cover the intended open description and required understanding for it's application. Some minor modifications are under development.

15.2 Syntax Elements

The following namebinding founds the superset of addressing attributes, which support specific addressing of explicitly one target as well as generic addressing of one or more targets by using search paths and content attributes, a.k.a. keywords or attribute value assertions.

The given sub-options are defined not to be order dependent, the keywords are case-insensitive.

The contained paranthesis, angle, and square brackets are just syntactic helpers, when they are part of the syntax, they will be quoted with single quotation marks.

```
<target-application-entity>:=<tae>
<tae>:=[<access-point>]<application>

<access-point>:=<physical-access-point>[<virtual-access-point>]


<physical-access-point>:=<pm>
<pm>:=<machine-address>[:<access-port>]

<virtual-access-point>:='['<vm>']'
<vm>:=<machine-address>[:<access-port>]


<application>:=<host-execution-frame><application-entity>
```

Figure 15.1: TAE - Target Application Entity address

```
<machine-address>:=
  [(ID|I|PATHNAME|PNAME|P):<mconf-filename-path>][,]
  [(BASEPATH|BASE|B):<base-path>[%<basepath>]{0,n}
  [(ID|I):<id>][,]
  [(LABEL|L):<label>][,]
  [(FILENAME|FNAME|F):<mconf-filename>][,]
  [(UUID|U):<uuid>][,]
  [(MAC|M):<MAC-address>][,]
  [(TCP|T):<TCP/IP-address>][,]
```

Figure 15.2: Machine-Address

```
<DISPLAYext>:=<target-display-entity>

<target-display-entity>:=<tde>
<tde>:=<tae>:<display>.<screen>
```

Figure 15.3: TDE - Target Display Entity address

The given general syntaxes lead to the following applied syntaxes with the slightly variation of assigned keywords.

```
<target-application-entity>:=<tae>
<tae>:=[<access-point>]<application>

<access-point>:=<physical-access-point>[<virtual-access-point>]


<physical-access-point>:=<pm>
<pm>:=<machine-address>[:<access-port>]

<virtual-access-point>:='['<vm>']'
<vm>:=<machine-address>[:<access-port>]


<application>:=<host-execution-frame><application-entity>
```

Figure 15.4: TAE - Target Application Entity address

The above minor variations take into account some common implementation aspects.

<access-point>:=<physical-access-point>[<virtual-access-point>]

The complete path to the execution environment.

<access-port>
The port to be used on the access-point.

<application>:=<host-execution-frame><application-entity>
The application itself, which has to be frequently used in combination with a given service as runtime environment.

<application-entity>
The executable target entity of the addresses application, which could be an ordinary shell script to be executed by a starter instance, or an selfcontained executable, which operates standalone within the containing entity. E.g. this could be a shared object or an executable.

The following extends the DISPLAY for seamless usage within ctys. So redirections of entities to any PM, VM of VNC session supporting an active Xserver will be supported. The only restrictions apply, are the hard-coded rejection of unencrypted connections crossing machine-borders.

```
TDE - Target Display Entity address
===================================

<DISPLAYext>:=<target-display-entity>

<target-display-entity>:=<TDE>
<TDE>:=<TAE>:display.screen
```

(basepath|base|b):<base-path>1,n
: Basepath could be a list of prefix-paths for usage by UNIX "find" command.

  When omitted, the current working directory of execution is used by default.

(filename|fname|f):<mconf-filename>
: A relative pathname, with a relative path-prefix to be used for down-tree-searches within the given list of <base-path>.

  So far the theory. The actual behaviour is slightly different, as though as a simple pattern match against a full absolute pathname is performed. Thus also parts of the fullpathname may match, which could be an "inner part". This is perfectly all right, as far as the match leads to unique results.

  More to say, it is a feature. Though a common standardname, where the containing directory of a VM has the same name as the file of the contained VM could be written less redundant, when just dropping the repetitive trailing part of the name.

<host-execution-frame>
: The starter entity of addressed container, which frequently supports a sub-command-call or the interactive dialog-access of users to the target system.

(id|i):<mconf-filename-path>
: The <id> is used for a variety of tasks just as a neutral matching-pattern of bytes, an in some cases as a uniqe VM identifier

within the scope of single machine. The semantics of the data is handled holomporphic due to the variety of utilized subsystems, representing various identifiers with different semantics. Thus the ID is defined to be an abstract sequence of bytes to be passed to a specific application a.k.a. plugin, which is aware of it's actual nature.

The advantage of this is the possibility of a unified handling of IDs for subsystems such as VNC, Xen, QEMU and VMware. Where it spans semantics from beeing a DISPLAY number and offset of a base-port, to a configuration file-path for a DomU-IDs, or a PID of a "master process".

This eases the implementation of cross-over function like LIST, because otherwise e.g. appropriate access-rights to the file are required, which is normally located in a protected subdirectory. These has to be permitted, even though it might not be required by the actual performed function.

(LABEL|L):<label>

```
<label>={[a-zA-Z-_0-9]{1,n} (n<30, if possible)}
```

User defined alias, which should be unique. Could be used for any addressing means.

(MAC|M):<MAC-address>
The MAC address, which has basically similar semantically meaning due to uniqueness as the UUID.

Within the scope of ctys, it is widely assumed - even though not really prerequired - that the UUIDs and MAC-Addresses are manual assigned statically, this could be algorithmic too. The dynamic assignment by VMs would lead to partial difficulties when static caches are used.

<mconf-filename>
The filename of the configuration file without the path-prefix.

<mconf-filename-path>
The complete filepathname of the configuration file.

<mconf-path>
The pathname prefix of the configuration file.

(PATHNAME|PNAME|P):<mconf-path>
When a VM has to be started, the <pathname> to it's configuration file has to be known. Therefore the <pathname> is defined. The pathname is the full qualified name within the callers namescope. SO in case of UNIX it requires a leading '/'.

<physical-access-point>:=<machine-address>[:<access-port>]
The physical termination point as the lowest element of the execution stack. This is the first entity to be contacted from the caller's site, normally by simple network access.

<target-application-entity>
The full path of the stacked execution stack, addressing the execution path from the caller's machine to the terminating entity to be executed. This particularly includes any involved PM, and VM, as well as the final executable. Thus the full scope of actions to be performed in order to start the "On-The-Top" executable is contained.

(TCP|T):<tcp/ip-address>
The TCP/IP address is assumed by ctys to assigned in fixed relation to a unique MAC-Address.

(UUID|U):<uuid>
The well known UUID, which should be unique. But might not, at least due to inline backups, sharing same UUID as the original. Therefore the parameter FIRST, LAST, ALL is supported, due to the fact, that backup files frequently will be assigned a name extension, which places them in alphabetical search-order behind the original. So, when using UUID as unique identifier, a backup will be ignored when FIRST is used.

Anyhow, cross-over ambiguity for different VMs has to be managed by the user.

<virtual-access-point>:=<machine-address>[:<access-port>]
The virtual termination point as an element of the execution stack. The stack-level is at least one above the bottom This stack element could be accessed either by it's operating hypervisor, or by native access to the hosted OS.

15.3 Stack Addresses

The stack address is a logical collection of VMs, including an eventually basic founding PM, which are in a vertical dependency. The dependency results from the inherent nested physical execution dependency of each upper-peer from it's close underlying peer. Therefore the stack addresses are syntactically close to GROUPS with additional specific constraints, controlling execution dependency and locality. Particularly the addressing of a VM within an upper layer of a stack could be smartly described by several means of existing path addresses schemas. Within the UnifiedSessionsManager a canonical form is defined for internal processing(Section 13.3.4 'Stacks as Vertical-Subgroups' on page 169), which is available at the user interface too. Additional specific syntactical views are implemented in order to ease an intuitive usage for daily business. The following section depicts a formal meta-syntax as a preview of the final ASN.1 based definition.

A stack address has the sytax as depicted in Figure 15.5.

```
<stack-address>:=<access-point-list>

<access-point-list>:=[<physical-access-point>|<virtual-access-point-list>]

<virtual-access-point-list>:=
       '['<virtual-access-point>']'['('<context-opts>')']
       [<virtual-access-point-list>]
```

Figure 15.5: Stack-Address

A stack can basically contain wildcards and simple regexpr for the various levels, groups of entities within one level could be provided basically to. And of course any MACRO based string-replacement is applicable. But for the following reasons the following features are shifted to a later version:

- Wildcards:
  An erroneous user-provided wildcard could easily expnad to several hundred VMs, which might be not the original intention. Even more worst, due to the detached background opera-

tion on remote machines, this can not easily be stopped, almost just ba reboot of the execution target. Which, yes, might take some time, due to the booting VMs.

- Level-Groups/Sets:
  Due to several highe priorities this version supports explicitly addressed entries only.

15.4 Groups of Stack Addresses

The usage of stacked addresses is supported for any entry, where an address is required, except for cases only applicable to PMs, e.g. WoL. The usage of stacked addresses within groups is supported too.

Therefore the behaviour for global remote options on ctys-CLI is to chain the option with any entity within the group, such as for the single PM case in Figure 15.6.

```
ctys -a <action> -- '(<glob-opts>)' <group>'('<group-opts>')'

=> group expansion results to:

   ...
   <group-member0>'(<glob-opts>)'('<group-opts>')'
   <group-member1>'(<glob-opts>)'('<group-opts>')'
   ...

=> host expansion result to:

   ...
   <group-member0>'(<glob-opts> <group-opts>')'
   <group-member1>'(<glob-opts> <group-opts>')'
   ...
```

Figure 15.6: Groups of Stack-Addresses

This behaviour of "chaining options" results due it's intended mapping to the internal canonical form before expanding it's options, to the permutation of the <group-options> to each member of the group. The same is true for the special group VMSTACK that the global and context options are in case of groups just set for the last - topmost - stack element Figure 15.7.

```
<group-member0>'(<glob-opts>)(<group-opts>)'

=> group expansion results to:

   '[<vm0>][vm1][vm2](<glob-opts>)(<group-opts>)'

=> host + stack expansion result to:

  level-0: <vm0>
  level-1: <vm0>'['<vm1>']'
  level-2: <vm0>'['<vm1>']''['vm2']''('<glob-opts>)'('<group-opts>')'
```

Figure 15.7: Groups member option expansion

When entries within the stack require specific context-options, these has to be set explicitly within the group definition, or the stack has to be operated step-by-step.

This behaviour is planned to be expanded within one of the next versions.

Chapter 16

CTYS Call interface

The most of preset default values could be changed by setting and exporting environment variables, e.g. PR_OPTS for "pr" options, or several require timeouts. These are documented inline within the sources.
Specific install and bootstrap information is provided in the README and INSTALL documents.

16.1 Common Elements and Behaviour

16.1.1 Sessions Namebinding

The current version basically binds the session to it's plugin type as an internal control identifier in order to use the appropriate funtion call, and addtional runtime data which is used to identify the actual instances. The identifying instance data such as LABEL has to be kept unique, because it is utilized within generic scans by means of context-free string comparison due to performance aspects.

16.1.2 Cache for Permormance and Collections

The name resolution for static and dynamic data could be performed basically by two approaches:

- from cashed data by ctys-vhost and related tools
- by polling of actual runtime data with "ctys -a LIST" call.

Both approaches have their advances and impacts as any cache system has.

When using the cache, this obviously has to be up to date, when a data match occurs. When no match could be get, the realtime

polling will be used by default.

The performance impact of the realtime poll vs. cached data is frequently a factor more than 10, which is 5-15 seconds vs. 0.5 seconds. The usual case is even more dramatic, usually 30 to 60 seconds, for CREATE, could be even much longer, depending from the file system to be scanned.

In cases of runtime systems scan via LIST it depends on the actual load. This approach is normally somewhat different from the pure CREATE call, because just a part of the data could be used from the cache DB, which just represents "potential candidates", the actual runtime state has to be polled in any

case, but could be done much more efficient of course.
But anyhow, in specific situations the usage of the poll approach is the only one applicable. The systems bootstrap for example is normally performed with an empty cache DB, so no decision is required. This is particularly the case, due to the usage of ENUMERATE itself for evaluation and collection of the cache DB.

The cache behaviour could be commonly controlled by the following two keyword:

NOCACHE Disables the usage of cached data for address resolution. Therefore the call of "ctys-vhost" and related utilities is disabled.

The default behaviour is to used CACHE first, which resolves for the low-medium reference system frequently within less than 0.5 seconds to a match. When no match found POLL is used, which is an internal ENUMERATE call and may have an performance impact of multiples of 10 and even more, depending on the size and structure of the scanned file system. The response time is frequently in the range of 30-60 seconds, and more.

When activated, at least polling has to be allowed, otherwise no resolution will be possible.

NOPOLL Disables the polling of data, thus CACHE is required. A missing hit on cached data will lead to failing of resolution.

16.1.3 Access VM Stacks

The utilization of VM stacks, which are nested and therefore encapsulate each level requires for several methods some specific recognition and stack-awareness.

Almost each action requires it's own specific handling.

CREATE The CREATE action is almost only effected specifically, when a stack-level is requested to be activated, for which the containing (n-1)-level stack entity is not yet present, this statement includes any lower missing entity too.

Therefore it has to be decided whether the request should be rejected or an implicit stack-recovery has to be performed.

When the decision is made for implicit recovery, the algorithm could be said to

`"process-upward-from-highest-existent".`

CANCEL The CANCEL action is effected by stack-awareness when a stack entity is requested to be canceled, which has active contained upper stack entities.

In this case the decision to be made are one or a combination of

- propagate CANCEL request to upper stack
- use hypervisor only, ignore upper stack
- "kill" if supported (e.g. not Xen) the current level hypervisor
- CANCEL the execution target, which is the execution level of stack "itself", e.g. essential for PMs.

When the decision is made for state-change-forward-propagation, the algorithm could be said to

1. "walk-upward-to-all-highest-existent"

2. "process-downward-to-request-target"

LIST This action has to be stack-aware on request, it is the most beneficial dynamic information collector for the user in order to manage VM stacks.

SHOW This action should be stack-aware on request, it is a beneficial dynamic information collector for the user in order to manage VM stacks.

ENUMERATE This action must not be stack aware, what is due to the practical circumstance, that the destination VMs has to be active, when to be enumerated for the executable VMs.

Thus, as long, as not the offline VM images could be scanned (which will be obviously never the case for all VMs), this call would have lead to the activation of ALL VMs executable, thus it probably could be said, will never finish due to bulk-start-performance-impact!

INFO Basically almost the same as ENUMERATE.

Due to the impact of stack awareness for execution, this behaviour could be controlled for the methods. Almost each action requires it's own specific handling.

FORCE Performs the requested action immediately by calling the hypervisor. No forward propagation is performed.

The specific range of features for the various hypervisors could and do vary, thus will be specialized within the description of the specific plugin.

STACK Performs the requested action first by state-propagation "deeper into the nested stack", which means upward due to ISO-like stack model.

The behaviour when failing with state-change-request propagation depends on the actual request.

The hypervisor might be called as final anchor in case the propagation fails or timeouts for a CANCEL request, because otherwise the request may fail, even though it could have been finished by usage of the hypervisor. A decision whether or not

to perform a "final and ultimate kill" on the stack is not obvious due to the hidden upper part, which basically could include any highly sensitive task, resulting in crucial failure when canceled abrupt. This risk will remain for generic actions, and has to be handeled by the application and it's close systems within the encapsulated stack entities.

For the CREATE action - which is in case of STACK an iterative upward execution - the call of the hypevisor is necessarily the first call on each level. Thus in case of CREATE in contrast to the CANCEL action the transparency of the stack is system immanent.

The call of the hypervisor could be followed by an state-change-propagation request after success only. The remaining question for partial execution is here what to do with an incomplete VM stack.

The answer for now is "let him live". This has to be monitored and handeled appropriately by the user when the "-b on" option for asynchronous background operations is choosen. In case of synchronous operations the job will be canceled, but also no roll-back is performed. Just the monitoring is implied due to synchronity.

The dynamic information poll methods LIST and SHOW just ignore a propagation failure, this is the same behaviour as in the case that nothing was found.

SELF The SELF keyword is currently for CANCEL action used only, but will of such basic level of understanding, that it is shown here.

SELF causes the inclusion of the execution target for requested operations too.

This is due to the technical requirement, that any method for an stacked entity has to be initially performed within it's container entity, which hosts the bottom controller instance of the targeted stack-level - it's hypervisor - for ultimate superposing action.

The basic call structure

```
ctys -a <action>=<action-target> <execution-target>
```

of ctys is splitted into the <execution-target> where the first communications peer entity resides, and the <action-target>, which is the destination of <action> to be performed.

The Stack Essentials - CREATE and CANCEL

One of the most beneficial features for stacked VMs would be the implicit creation of missing sessions within the execution stack. Anyhow, even though it is basically possible to do this with some current VM implementations, this feature is shifted.

The reason is simply the lack of a common strategy within the used VMs, where at least XEN and QEMU would be suitable, but others not for an unattended and SERVERONLY iterative CREATE chain. In avoidance of a quick-shot resulting in some "savage-variants" when coming to details, the decision for this version is not to implement the automatic and implicit creation of inactive intermediate stack entities.

Therefore this feature is currently shifted and will be implemented within a later version. For now each stack entity has to be CREATED seperately and explicitly by the user, in manual iteration.

For CANCEL anything stays as it is, stacked operations are supported as described.

16.2 ctys Actions

The following actions are supported:

Handling of addressed sessions:

CREATE: Start and/or connect to sessions

CANCEL: Finish, suspend and kill sessions

Retrieve overall generic information:

ENUMERATE: static information about stored VMs

INFO: static details about ENABLED VMs

LIST: dynamic information about running VMs

SHOW: dynamic details about ENABLED VMs

Internal helper-methods using the official interface:

GETCLIENTPORT

CANCEL

```
CANCEL=(<machine-address>)\{1\}|ALL
  (
    [FORCE|STACK][,]
    [SELF][,]
    [
      RESET
      |REBOOT
      |(INIT:<init-state>)
      |(PAUSE|S3)
      |(SUSPEND|S4)
      |((POWEROFF|S5)[:<timeoutBeforeKillVM>]
    ][,]
    [(CLIENT|SERVER|BOTH)][,]
    [TIMEOUT:<timeout-value>]
    [,USER:<user>[%<credentials>]]
  )
```

<machine-address>
Refer to common options parts description.

ALL
Cancels all instances of actually loaded plugins collected by a list call.

Therefore it should be considered, that implicitly loaded plugins are treated too, even if some instances might be present, which ar utilized standalone. This could effect e.g. native VNC HOSTs sessions in case of XEN and QEMU, where the VNC viewer is required as the CONSOLE.

This behaviour might change.

CLIENT|SERVER|BOTH Selection of the VM parts to be CANCELed. The only practical applicable distinction for

ctys might be CLIENT or else. This is due to the case BOTH handles all anyhow, and the temination, even temporary unavailability for RESET, terminates the CLIENT implicitly for any current supported plugin.

FORCE
: Cancels the stack by call to hypervisor. No forward propagation is performed. If supported, a kill call is performed after a timeout.

STACK
: Cancels stack by forward propagation of CANCEL request. If supported, a kill call is performed after a timeout.

SELF
: Includes the execution target as final system to be canceled. Else the upper stack beginning with the <machine-address> is canceled only. If no upper stack is present nothing will be done.

REMARK:
: Due to it's possible consequences the SELF parameter is NOT SET by DEFAULT, thus has to be provided explicitly.

  For variation refer to the specific plugin.

RESET
: Resets the target, for a stack element an appropriate forward propagation of the RESET request will be performed. Therefore a mapping as given in the following description is proceeded.

  RESET utilizes the native call of the hypervisor as a second step when FORCE is not set.

REBOOT
: Reboots the target, the behaviour is similar to RESET, as far as the first step is a stack propagation controlled by FORCE flag. The difference here is the lack of the usage of a hypervisor as second step. Thus a REBOOT call with FORCE flag set and SELF flag unset will not perform any CANCEL action, whereas RESET calls the hypervisor as the only action. A REBOOT call without FORCE flag set and SELF flag unset will perform a stack propagation, and thus in case of missing native GuestOS support for ctys

the hypervisor will be called by the propagation function.

The support of GuestOSs requires a proper setup of ctys within the GuestOS and preferably the inclusion of required access data registered within the cacheDB.

INIT:<init-state>
: Performs an native INIT on UNIX systems. Therefore an appropriate forward propagation by remapping of the INIT request is performed as described in the following.

PAUSE|S3
: Calls PAUSE when supported, which is an ACPI state S3 - Suspend To RAM. Therefore the call will be propagated and first requested to GuestOSs within the stack, else the hypervisor of the current level is called.
When propagation is not appreciated, the FORCE flag should be used.

SUSPEND|S4
: SUSPEND has almost the same behaviour as PAUSE, the only difference is the state S4 - Suspend To Disk.

POWEROFF|S5[:<timeoutBeforeKillVM>])
: POWEROFF or ACPI state S5, switches into offline mode, which is actually a stand-by mode. Within UnifiedSessionsManager the configuration of WoL is pre-required in order to activate systems from state S5.
The support of remote power switches is foreseen for a future version.

TIMEOUT:<timeout-value>
: This is the timeout after forward propagation of a CANCEL request for each of contained current-level-entries. The values are set independently.
After the timeout the next escalation level will be performed.

USER:<user>%[<credentials>]
: The user to be used for native access to the <action-target>. Default user and used authentication method is the same as used for authentication on the execution target.

Additional information is available in the common options section for USER .

The cancel method requires the adaption of parts of the generic superset to plugin type specific addressing schema. This is due

to the various fields of application, so e.g. a VM could be addressed by its ID, which is a fully qualified pathname in most cases, and will be used as access key by various VMs. The ctys tools extend this schema e.g. for usage of the UUID in combination with a provided base-path for automatic scanning and remapping to the VMs ID, Label and UNIX-pid.

This of course is not applicable to arbitrary VNC sessions, even though the hosting machine should have the same information, and though a UUID will basically be available. Multiple VNC or X11-sessions on the same host can not be distinguished by a UUID only.

When handling stacks of nested PMs and VMs, the application of CANCEL action on a lower level will naturally force contained instances to be terminated too. Thus a behaviour has to be defined, whether a top-down soft shutdown has to be performed, or a "bottom-up" behaviour of instances, by killing the assigned level without recognition of contained instances. This might be appropriate e.g. in emergency cases.

Two basic directions are defined:

FORCE
: As described above.

STACK
: Uses a chained approach for shutting down by an top-down behaviour. Therefore the VM-stack will be first walked up and marked by repetitive sub-calls with defined specific CANCEL suboptions. These suboptions will be applied top-down in a roll-back-chain once reaching the top of the stack.

  Due to implementation specifics some remapping of inner states is performed, for additional information refer to Section 8.4.3 ‘Shutdown’ on page 99 , particlarly to the table containing the mapping for State Propagation.

The internal parameter SELF controls the VM stack to be CANCELed. Due to the requirement of calling CANCEL on the hypervisor itself, the <execution-target> receiving the CANCEL job is in case of a VM not the <cancel-target>, thus not

to be canceled itself. So these entity propagates the CANCEL job to the stack and sets the SELF flag for all following entities to cancel theirself if applicable, else the hypervisor will force it.

The exception it a PM, which has no lower container. Therefore a PM as the recipient of a CANCEL job will cancel itself in any case. Similar exception applies for a PM to be CREATEd while it is not yet running.

The mode of operation is controlled by following parameters. When not given, the default CANCEL mode is applied, which is a SHUTDOWN/STOP for servers and an UNIX-kill for client processes.

INIT:<init-state>
: Mapped to UNIX init.

PAUSE
: The VM will be paused immediately, remaining clients will be "UNIX-killed", not so if SERVER selected.

SUSPEND
: The VM will be suspended immediately, remaining clients will be "UNIX-killed", not so if SERVER selected.

RESET
: A reset is performed on any instance immediately, remaining clients will be "UNIX-killed", not so if SERVER selected.

REBOOT
: Performs a "soft reset", where the instances will be given a timeout before forcing them to terminate.

POWEROFF[:<timeoutBeforeKillVM>]
: A multilevel-delayed SHUTDOWN of VMs is performed.
CANCEL works by default asynchronously, because it should be usable for emergency shutdown, where a bunch of sessions on a server has to be killed immediately more or less accurate. So give all of them same delay and chance.
When in case of required proper stack operations synchronous operations is required, a specific timeout could be provided. After the exhaust of the timeout, the target item will be just killed by means of containing OS. The option "-b" controls the common behaviour, whereas specific methods supply variations.

CREATE

```
CREATE=<machine-address>\{1\}
[
  CONNECT
  |REUSE
  |RECONNECT
  |RESUME
][,]
[USER:<user>[%<credentials>]][,]
[CONSOLE:<console-type>][,]
[BOOTMODE:<boot-mode>][,]
[PING:(OFF|<#repetition>%<sleep>)][,]
[SSHPING:(OFF|<#repetition>%<sleep>)][,]
[STACKCHECK:<stack-check>][,]
[WAITC:<timer>][,]
[WAITS:<timer>][,]
[<callopts>][,]
[<xopts>]
```

Start remote session. Session type specific parameters could/have to be provided. The given sub-options are not order dependent, the keywords are case-insensitive. For call details refer to the specific package.

<machine-address>
Refer to common options parts description.

BOOTMODE

```
BOOTMODE:
  (
    KERNEL
    |PXE
    |FDD
    |CD
    |vHDD
    |ISO
    |INSTALL
  )
```

The BOOTMODE parameter supports the alteration between pre-configured boot setups. Due to various levels of capabilities of the different hypervisors, the support for this suboption differs between the supported hypervisors.

The usage of this parameters requires a wrapper script specific to the UnifiedSessiosnManager or a specific configuration variant to be in place. This is optional for XEN , whereas it is mandatory for QEMU anyway. For supported VMW products in this version the bootmode has to be configured by utilization of the emulated motherboard BIOS.

The most complete support is provided by QEMU , which could handle an almost exhausting number of variants by it's own command line parameters. Thus the wrapper itself could provide any adaption.

For XEN the parameters are implemented in this version as an standard extension to a specific pre-configured vmx file.

XEN-Mode	vmx-name
CD	<vmx-name>-cd.conf
FDD	<vmx-name>-fdd.conf
USB	<vmx-name>-usb.conf
ISO	<vmx-name>-iso.conf
KERNEL	<vmx-name>-kernel.conf
PXE	<vmx-name>-pxe.conf
VHDD	<vmx-name>.conf
INSTALL	<vmx-name>-inst.conf

Table 16.1: Supported Boot-Modes for XEN

In case of XEN the files could be supported either by their fully qualified pathname suboption including the specific extension without usage of the BOOTMODE. Alternatively the base could be addressed by common means of <machine-address> in companion with the usage of BOOTMODE, where the required name is constructed by combining both.

It is recommended for the XEN module to set the "MAGICID-XEN" in the default file only, and to set "MAGICID-IGNORE" in the others. This controls the inclusion into the cacheDB, which should be neatless with the other plugins, even though has to be remembered than. Some ambiguity may occur too when not disabled for caching.

The next table lists the various support options for current base-plugins.

Mode	QEMU	VMW	XEN
CD	X	(BIOS)	X
FDD	X	(BIOS)	X
USB	X	(BIOS)	X
ISO	X	(BIOS)	X
KERNEL	X	-	X
PXE	X	(BIOS)	X
VHDD	default	default	default
INSTALL	X	(BIOS)	X

Table 16.2: Supported Boot-Modes

- CD
  Physical CD/DVD located on the execution target.
- FDD
  Physical FDD located on the execution target.
- ISO
  An ISO image file accessible on the execution target.
- KERNEL
  An kernel and initrd image file accessible on the execution target. This is specific to Linux.
- PXE
  Network boot, requires preconfigured DHCP, TFTP, and HTTP/FTP/NFS.
- vHDD
  The virtual HDD file accessible on the execution target, basically similar to ISO, but with different format. This is the default, where vHDD is the virtual install target.
- INSTALL
  The install version is called.

CONNECT
: Connects to an existing session, else an exit with error state is performed. Therefore a new client is started.

CONSOLE

The CONSOLE supports a common user access facility, which could be an ordinary CLI interface to be used within a shell, as well as a VNC based remote desktop accesible from a X11 based local desktop. Several additional console types are planned to be added.

The main distinction between a CONSOLE and the HOSTs access is the direct and native access of a CONSOLE to a hypervisor facility, whereas a HOSTs session is an ordinary full-scale login into the running GuestOS.

Anyhow, due to practical considerations some "soft-extend" to the basic principle was made. So the HOSTs types theirself provide some CONSOLE. The PM is due to generic handling based on "soft-consoles" only, but some RS232-Terminal-Server and Remote-LAN-Server facilities will follow.

In addition to the currently supported explicit choices, almost any type of local or remote frontend could be provided by combining DISPLAYFORWARDING / CONNECTIONFORWARDING and an explicit call by CMD suboption of CLI or X11 plugin. For examples refer to Section 21.1 ‘CLI Examples’ on page 429 and Section 21.2 ‘X11 Examples’ on page 437 .

Anyhow, for now the following console types are supported for the various plugins as preconfigured enumerations.

CONSOLE-Type	Plugins						
	QEMU	VMW	XEN	PM	VNC	X11	CLI
CLI[1]	X	-	X	X	-	-	X
EMACS[2]	X	-	X	X	-	X	-
EMACSM[3]	X	-	X	X	-	(X[4])	-
EMACSA[5]	X	-	X	X	-	X	-
EMACSAM[6]	X	-	X	X	-	(X[4])	-
GTERM[7]	X	-	X	X	-	X	-
NONE/initHeadless	X	(*[8])	X	X	X	-	-
SDL	X[9]	-	(*[10])	-	-	-	-
VNC	X	(X[11])	X	X	X[12]	-	-
VMW	-	X [13]	-	-	-	-	-
XTERM[14]	X	-	X	X		X	

Table 16.3: Supported Console-Types

[1] CLI is supported as interactive console in combination with modes "-b 0,2" and "-z 2" only .

[2] EMACS is "shell-mode", for now no ctys-coloring is supported. Anyhow, the terminal is a full scale buffer, thus any editing functionality is available. Many, many thanks to the authors of Emacs, it's just great.

[3] EMACSM supports a multiple-window stack-view. The upper-window opens a CLI session to the GuestOS, the lower to the PM.

[4] For the X11-only case the content of the primary window has to be provided.

[5] EMACSA is "ansi-term", supports colors, but limited edit functionality. Execution of commands is not supported.

[6] EMACSAM is "ansi-term" similar to EMACSM.

[7] GTERM is the "gnome-terminal".

[8] Found "nogui" option for usage with vmrun, will be introduced soon.

[9] Server resolution is supported for serial devices only by "-serial vc:..." option of QEMU, has to be configured with configuration wrapper.

[10] Will be considered to be supported.

[11] Supported for WS6 only by "hard-configured" ports, some mouse and keymap challanges.

[12] Support of "-r" option for independent server resolution, when missing implicitly set to same as client, see "-g" .

[13] Some specifics due to included MuxDemux for connections via a single port(default 904). Particularly CONNECTIONFORWARDING has it's own behaviour for remote login, but works. Additional constraints for VMPlayer: communications facilities(no C/S, no headless), and "-g" parameter(no geometry). For WS and Server the "-r" option for server resolution requires VM options to be set, refer to plugin-help.

[14] Yes, you'r right.

The following plugin types support the execution of an arbitrary command.

CONSOLE-Type	Plugins						
	QEMU	VMW	XEN	PM	VNC	X11	CLI
CLI	-	-	-	-	-	-	X
EMACS	-	-	-	-	-	X	-
EMACSA	-	-	-	-	-	-	-
GTERM	-	-	-	-	-	X	-
NONE/initHeadless	-	-	-	-	-	-	-
VNC	-	-	-	-	-	-	
XTERM	-	-	-	-	-	X	

Table 16.4: Supported Command-Execution

DEBUG
This feature is currently under study and development.

DEBUG-Type	Plugins						
	QEMU	VMW	XEN	PM	VNC	X11	CLI
DDD	*	-	*	*	-	*	*
ECLIPSE	*	*	*	*	-	-	-
GDB	X	-	*	*	-	*	*
MON	*	-	-	-	-	-	-
NONE/initHeadless	X	-	*	*	-	-	-

Table 16.5: Supported Debugger

PING:(ON|OFF|<repetition>%<sleep>)
Controls whether access to the CREATE target should be verified before termination of current task. This is on and set to default values by default. The usage is mandatory for stacked operations, where obviously contained entities required their container to be present.

The PING could be either switched off, or modifie in it's behaviour, by adjusting the number of repetitions and the sleep-period between each.

PING is used as first to verify accesibility of the TCP stack, before an eventual actual access is checked by SSHPING.

This is mandatory required, when JOBDATA should be stored and visible by LIST action. Thus the OFF value should not be used. For changed default to OFF, the ON value is available.

REUSE

Basically the same as CONNECT. When matching REUSE keyword, first it is tried to find matching sessions and if found, this will be connected instead of creating a new one. If missing, a new session is created. REUSE fails if already a client is attached to the server of that session. Therefore use RECONNECT if this is required. ...

...On Workstation, not so on Server. But anyhow, this depends on several factors, and at the end of the day probably on the users configuration too.

Thus REUSE just checks whether a server/session-entity is already running. The influence of already running clients to the success of call is determined by the actual product with its current configuration.

For the creation of unique non-shared sessions refer to RECONNECT.

RECONNECT

Basically the same as REUSE, with the difference, that any client session will be terminated before a new ONE is established. Additional could be opened with REUSE or CONNECT in shared-mode.

When RECONNECT sub-option is given, any previously running client(and only the clients!) will be canceled before starting the new client. This could be restricted by assigning access-rights to any of current clients, which has to be handled by underlying security layer. Therefore it is essential for any VM to be configured as previously mentioned - to continue running the server in the background, when the client is detached. OTHERWISE this will FAIL AND the SERVER TERMINATES.

This option fits pretty well, if a pure repositioning or resizing of the current client has to be performed, whereas the server, thus the "virtual screen output" of the running server remains unchanged and continues meanwhile so as

if a screen is still attached.

Due to access and locality restrictions following cases of "-L" option(including DEFAULT) has to be distinguished. The several permutations with process owners USER and GROUP are not handeled here. In general could be said, access rights are prerequisites from underlying security layer has to be guaranteed.

DISPLAYFORWARDING
: This is the most simple case, where by definition any client is running coallocated with its server, so could be simply found, canceled and restarted locally.

CONNECTIONFORWARDING
: This becomes some what tricky already:

- Previously executed clients running on the callers machines. Almost as simple as DISPLAYFORWARDING, if both are managed by ctys, of course.
- Previously executed clients are running on a machine different from current caller's machine. In current implementation this is not supported.

SERVERONLY
: Not applicable.

LOCALONLY
: FFS.

RESUME
: Resumes a previously suspended session. This can differ between the various plugins.

SSHPING:(ON|OFF|<repetition>%<sleep>)
: Controls whether access to the CREATE target should be verified before termination of current task. This is on and set to default values by default. The usage is mandatory for stacked operations, where obviously contained entities required their container to be present.

The SSHPING could be either switched off, or modifie in it's behaviour, by adjusting the number of repetitions and the sleep-period between each.

Before the SSHPING a PING is utilized in order to preassure basic TCP access. The SSHPING additionally just calls an "echo" command in order to verify the complete

SSH access path.

For changed default to OFF, the ON value is available.

STACKCHECK:<stack-check>
The STACKCHECK attribute defines and/or deactivates specific pre-checks for the current VM when used in a VM-STACK context. These values could be commonly predefined within the configuration file of each plugin individually and reversed by incident when used. The following attributes could be set individually.

```
STACKCHECK:
  OFF
  |
  (
   [(CONTEXT|NOCONTEXT)][%]
   [(HWCAP|NOHWCAP)][%]
   [(STACKCAP|NOSTACKCAP)]
  )
```

for additional information refer to STACKCHECK on VMSTACK and to Section 8 'PMs and VMs - The Stacked-Sessions' on page 81 .

USER:<user>%[<credentials>]
The user to be used for native access to the <action-target>. Default user and used authentication method is the same as used for authentication on the execution target.

Additional information is available in the common options section for USER .

WAITC:<timer>
Initial "sleep <timer>" after execution of client, once performed before an eventually PING and/or SSHPING.

The effect is visible for the user when operating in SYNCHRONOUS mode only, else may have internal influence only. The same is true for "detaching" components, which have actually to be polled for emulation of synchronous operations. This has to be done for some plugins/components even in order to detect the final execution state, where the call interface returns an OK status for it's own execution only.

WAITS:<timer>
Initial "sleep <timer>" after execution of server, once performed before an eventually PING and/or SSHPING.

The effect is visible for the user when operating in SYNCHRONOUS mode only, else may have internal influence only. The same is true for "detaching" components, which have actually to be polled for emulation of synchronous operations. This has to be done for some plugins/components even in order to detect the final execution state, where the call interface returns an OK status for it's own execution only.

<callopts>
Refer to common options parts description.

<xopts>
Refer to common options parts description.

ENUMERATE

```
ENUMERATE
[=
  (
    (
      (
        [ARCH][,]
        [CATEGORY|CAT][,]
        [CONTEXTSTRING|CSTRG][,]
        [CTYSRELEASE][,]
        [DIST][,]
        [DISTREL][,]
        [EXECLOCATION][,]
        [GATEWAY][,]
        [HWCAP][,]
        [HWREQ][,]
        [HYPERREL|HYREL][,]
        [IDS|ID][,]
        [IFNAME][,]
        [LABEL|L][,]
        [MAC|M][,]
        [NETMASK][,]
        [OS][,]
        [OSREL][,]
```

```
            [PLATFORM|PFORM][,]
            [PM|HOST][,]
            [PNAME|P][,]
            [RELAY][,]
            [RELOCCAP][,]
            [SERIALNUMBER|SERNO][,]
            [SERVERACCESS|SPORT|S][,]
            [SSHPORT][,]
            [STACKCAP|SCAP][,]
            [STACKREQ|SREQ][,]
            [STYPE|ST|TYPE][,]
            [TCP|T][,]
            [USERSTRING|USTRG][,]
            [UUID|U][,]
            [VCPU][,]
            [VERSION|VERNO|VER][,]
            [VMSTATE|VSTAT][,]
            [VNCBASE][,]
            [VNCDISPLAY|DISP][,]
            [VNCPORT|CPORT][,]
            [VRAM][,]
          )
          [TITLE|TITLEIDX|TITLEIDXASC][,]
          [MACHINE|MAXKEY][,]
        )
        | TAB_GEN:<tab-args>
      )
      [IP|DNS][,]

      [,MATCHVSTAT:
        (
          ACTIVE|TEMPLATE|BACKUP|TESTDUMMY
        )
        [ACCURATE]
        [,ACCURATE]
        [(ALL|EMPTY|PRESENT)]
      ]

      [,TERSE]
      [,PKG:<pkglist>]
      [,SORT[:[ALL|EACH][%UNIQUE][%<sort-key>]]]
      [,(BASEPATH|BASE|B):<base-path>[%<base-path>]{0,n}]
    ]
```

Enumerates all available remote Sessions with statically stored configurations. The enumeration is performed for all loaded plugins on the target machine. Candidates for the ENUMERATE action are memberes of the categories PMs and VMs. The results could be used for a CREATE action. If basepath# is supplied the following find operation is performed: "find $basepath1 $basepath2 ... -name '*.vmx' -print" else "$HOME" is used as base for find command.

The output is as listed in the following record description. Some exceptions occur, when multiple interfaces are configured within a VM. Each interface is assigned with each of it's IP address to a seperate output record, containing a single MAC address and a single assigned TCP address. Thus the number of output records is increased for multihomed VMs and PMs.

Some additional values are supported for basic management of VMs by simply adding masked keywords to present configuration files and/or directories. For additional description refer to Section 33.7 'Configuration Entries for ctys' on page 624 .

When "-X" option is set, the output is prepared as ";" semicolon seperated list for post-processing. The same is true, when setting TERSE.

Processing-Key	Short Description
ALL	Sets the output to a superset of valid fields, where some unused or reserved fields are suppressed.
DNS	Transforms TCP addresses to numeric format.
IP	Transforms TCP addresses to numeric format.
MACHINE	Sets the output to the canonical format which contains absolutely any field.
MATCHVSTAT	Alters the VMSTATE attribute, to be semantically matched.
MAXKEY	Sets the output to common subset.
PKG:<pkg-list>	Constrains on output to defined list.
SORT	Activates sort filter.
TAB_GEN	Activates table filter.
TERSE	Output for post processing.
TITLE	Output of field names.
TITLEIDX	Output of field names with indexes.
TITLEIDXASC	Output of field names with indexes and additional ASC-II column indexes for spreadsheet display.

Table 16.6: Processing suboptions

Nr.	Field-Key	Fieldname	Common
1	PM|HOST	ContainingMachine	X
2	TYPE	SessionType	X
3	LABEL|L	Label	X
4	ID	ID	X
5	UUID	UUID	X
6	MAC	MAC	X
7	TCP	TCP	X
8	DISPLAY	DISPLAY	
9	CPORT	ClientAccessPort	
10	SPORT	ServerAccessPort	
11	VNCBASE	VncBasePort	
12	DIST	Guest-Distro	
13	DISTREL	The release of the distribution.	
14	OS	Guest-OS	
15	OSREL	OS-Release	
16	VERNO	VM-Config version number	
17	SERNO	VM-SerialNo	
18	CATEGORY	Category	
19	VMSTATE	The state of the VM	X
20	HYPERREL	Release of the hypervisor used for installing the VM.	X
21	STACKCAP	The capabilites supported.	
22	STACKREQ	The list of capabilites required.	
23	HWCAP	Offered virtual HW.	
24	HWREQ	Required HW, either virtual or physical.	
25	EXECLOCATION	Defines the possible execution locations.	
26	RELOCCAP	Defines LOCATION behaviour.	
27	SSHPORT	Alternative port for "-p" option of SSH.	
28	NETNAME	Network name of the TCP/IP of current interface.	
29	RESERVED07	For future use.	
30	RESERVED08	For future use.	
31	RESERVED09	For future use.	
32	RESERVED10	For future use.	
33	IFNAME	The name of the interface within the GuestOS.	
34	CTYSRELEASE	MAGICID of the originator for each record.	
35	NETMASK	Internet NETMASK.	
36	GATEWAY	Internet Gateway.	
37	RELAY	The interconnection interface.	
38	ARCH	Virtual architecture presented to the GuetsOS.	
39	PLATFORM	Virtual device.	
40	VRAM	The pre-configured amount of RAM.	
41	VCPU	The pre-configured number of V-CPUs.	
42	CONTEXTSTRG	A private context storage for the plugin	
43	USERSTRING	A string to be customized by the user.	

Table 16.7: Output Record-Format for MACHINE suboption

GETCLIENTPORT

```
GETCLIENTPORT=<label>|<id>{1}
```

Returns the port for attaching the front end client services to the server component. This will be used internally only, or within plugins and macros. Security is based on SSH for ctys execution and the appropriate options of the current VM for restricting to local access only.

The output is presented as follows:

```
"CLIENTPORT(<type,<FQDN-host>,<vm-label>)=<client-access-port>"
```

Which could be for example:

```
"CLIENTPORT(VMW,host01.fantasy,linuxBox)=904"
```

INFO

Displays several static information for the given hosts. This action is currently under development, thus might change soon. Currently some OS and Machine information is displayed. Particularly the HW-Virtualization flags are shown. For now the Display is given as:

```
bash-3.1\$ ctys -a info -W delphi
############################
Node:delphi.soho
System      :Linux
OS          :GNU/Linux
RELEASE     :2.6.21.6-delphi-005
MACHINE     :i686
KERNEL#CPU  :SMP-KERNEL
CPU-INFO
processor:0
vendor\_id           :GenuineIntel
cpu family          :6
model               :11
model name          :Intel(R) Pentium(R) III CPU ...
stepping            :4
cpu MHz             :1266.131
cache size          :512 KB
processor:1
```

```
vendor_id           :GenuineIntel
cpu family          :6
model               :11
model name          :Intel(R) Pentium(R) III CPU ...
stepping            :4
cpu MHz             :1266.131
cache size          :512 KB

Flags assumed equal for all processors on same machine:
flags
vmx(VT-x - Pacifica)    = 0
svm(AMD-V - Vanderpool) = 0
PAE                     = 1

MEM-INFO
MemTotal               :  4018 G
SwapTotal              : 24579 G

VNC         :VNC Viewer Free Edition 4.1.2 for X - ...
wmctrl      :wmctrl is on this machine not available

------------
ctys:       :01\_02\_003a10
Plugings:   :   VNC
```

LIST

```
LIST[=
  (
    (
      (
        [ (TUNNEL) |(CLIENTS|C)|(SERVER|S)|(BOTH|B)][,]

        [CPORT][,]
        [DISPLAY][,]
        [GROUP|GID][,]
        [ID][,]
        [(JOBID|JID)][,]
        [LABEL][,]
        [MAC][,]
        [PID][,]
        [PM|HOST][,]
```

```
          [PNAME|P]
          [SPORT][,]
          [TCP][,]
          [TYPE][,]
          [UUID][,]
          [USER|UID][,]
        )
        [TITLE|TITLEIDX|TITLEIDXASC][,]
        [MACHINE|MAXKEY][,]
      )
      | TAB_GEN[:<tab-args>][,]
    )
    [IP|DNS][,]
    [,SORT[:[ALL|EACH][%UNIQUE][%<sort-key>]]][,]
    [PKG:<pkg-list>][,]
    [TERSE][,]
    [USER:<user>[%<credentials>]][,]
]
```

Lists active local and remote sessions. All loaded types are listed by filtering according to results of given filter assembled by provided suboptions.

The base set to be filtered is defined by the options "-t" and/or "-T". If "-t" is not present, the default "-t ALL" will be applied to all pre loaded plugins. For changing the selection scope of listed users refer to "-s" option.

The LIST action is deeply influenced by the setting of the option "-b" concerning the perfomance, and the option "-C" concerning the way the output data is displayed.

The basic influence on the display is described in Section 6.4 ‘Parallel and Background Operations’ on page 74 , the perfomance repercussion is presented in Section 27 ‘Performance Measures’ on page 555 .

LIST supports three display modes, where the displayed subset of fields could be configured by switching on with the assigned keyword.

1. tables

The tables mode supports semi-fixed and generic tables.

- semi-fixed tables
  These tables have a fixed number of displayed columns with limited content variation, but fully resizable column-widths.

  The most important view is the TAB_TCP, which shows the columns

  - TCP-Container: PM
  - TCP-guest: (MAC|TCP|DNS)
  - Variable:
    Label: label
    ID: ids|id
  - Sesstype: (PM|CLI|X11|VNC|VMW|XEN|QEMU)
  - C: (C|S)
  - User: $USER
  - Group: <group>

  The sizes could be defined by providing an integer width for each column seperated by "%". Contents will be truncated righthand when they extend the size of the column.

  "TAB_TCP:7%%6%%3%%"

  This defines the following sizes:

  - TCP-Container(1): 7 (default=17)
  - TCP-guest(7): default (default=17)
  - Label(3): 6 (default=20)
  - Sesstype(2): default (default=8)
  - C(14): 3 (default=1)
  - User(12): default (default=10)
  - Group(13): default (default=10)
- generic tables
  Generic are completely customizable.

Each table has to contain at least one mandatory column. The number, order, and repetition is free to be defined. But no semantics check or display format validation is performed on the results.

The resulting syntax is not very handy for frequent manual application, thus the advantage is given when prepared configurations are stored within macro files for repetitive application.

The syntax is as follows.

TAB_GEN:<idx>_<colname>_<width>[%<idx>_<colname>_<wid

The underscore "_" is here reserved as field seperator, thus could not be used within regular values. Enclosed empty fields are not allowed.

For each column the triple of data is required:

<idx>
Canonical index of field to be displayed. The value could be evaluated by calling list with the "titleidx" option.

<colname>
The arbitrary name of the column to be displayed. Following restrictions apply:
- No spaces
- Printable characters only
- Size will be cut, when extends the column width.

<width>
The width of the column, this excludes the border marker, which will has to be added for size calculations. each file has one additional marker, minus one from the total sum.
The number of columns is "unlimited", has to fit the screen, of course.

2. "nonterse"
Displays a listing, which partly "terse", but listed with some structure and ordering for different targets.

3. "terse"
   Displays a listing, which is a neatless sum of selected fields for each record on one line, with fileds seperated by a semicolon ";". This format could be imported to various office tools unchanged. When the option "MACHINE" is provided additionally, the full set of information is listed. Non-present values are just displayed as empty fields.

Nr.	Field-Key	Fieldname	Common
1	PM|HOST|H	ContainingMachine	X
2	TYPE	SessionType	X
3	LABEL|L	Label	X
4	ID|I|PNAME|P	ID	X
5	UUID	UUID	X
6	MAC	MAC	X
7	TCP|T	TCP	X
8	DISPLAY	DISPLAY	
9	CPORT	ClientAccessPort	
10	SPORT	ServerAccessPort	
ffs	VNCBASE	VncBasePort	
11	PID	PID	
12	UID	UID	
13	GID	GID	
14	CSTYPE	C/S-Type	
15	JOBID	JobID	

Table 16.8: Output-Format for MACHINE suboption

Processing-Key	Short Description
DNS	Transforms TCP addresses to numeric format.
IP	Transforms TCP addresses to numeric format.
MACHINE	Sets output to canonical format.
MAXKEY	Sets output to common subset.
PKG:<pkg-list>	Constrains on output to defined list.
SORT:[:<sort-args>]	Activates sort filter
TAB_GEN:<tab-args>	Activate table filter
TERSE	Output for post processing.
TITLE	Output of field names.
TITLEIDX	Output of field names with indexes.
TITLEIDXASC	Output of field names with numerical and additional ASC-II column indexes for spreadsheet display and canonical fields .

Table 16.9: Processing suboptions

SHOW

Displays dynamic information for the given hosts. This action is currently under development, thus might change soon. Currently some OS and Machine information are displayed. Particularly a short resource-analysis for manual load-distribution by rule of thumbnail is shown. In addition the "ALARM" keywords of lm-sensors are "grepped" and displayed, when some occurs. This is due to avaoidance of new VM starts in case of some existing ALARMS. The package lm_sensors therefore has to be installed and configured.

For now the Display is given as:

```
bash-3.1\$ ctys -a show -W delphi
############################
Node:delphi.soho
System      :Linux
OS          :GNU/Linux
RELEASE     :2.6.21.6-delphi-005
MACHINE     :i686
MEM-INFO
MemTotal            :  4018 G
MemFree             :    96 G
SwapTotal           : 24579 G
SwapFree            : 24579 G
Top         : iterations=10
top - 10:25:33 up 1 day, 12:39,  1 user,  load aver...
Tasks: 241 total,   1 running, 240 sleeping,   0 st...
Cpu(s):  0.4\%us,  0.4\%sy,  0.0\%ni, 98.6\%id,  0.5\%wa...
Mem:   4018724k total,  3922936k used,    95788k fr...
Swap: 24579420k total,        4k used, 24579416k fr...

PID USER      PR  NI  VIRT  RES  SHR S \%CPU \%MEM ...
28241 vadmin     5 -10  388m 308m 296m S    0  7.9 ...
28246 vadmin     5 -10  388m 308m 296m S    0  7.9 ...
28247 vadmin     5 -10  388m 308m 296m S    0  7.9 ...
28248 vadmin     5 -10  388m 308m 296m S    0  7.9 ...
28249 vadmin     5 -10  388m 308m 296m S    0  7.9 ...
28250 vadmin     5 -10  388m 308m 296m S    0  7.9 ...
28251 vadmin     5 -10  388m 308m 296m S    0  7.9 ...
```

```
28252 vadmin     5 -10  388m 308m 296m S    0  7.9 ...
28253 vadmin    15   0  388m 308m 296m S    0  7.9 ...
28230 vadmin    15   0 99216  32m  16m S    0  0.8 ...
3640 root      15   0 41332  27m 3596 S    0  0.7 ...
23299 acue      15   0 45984  24m  14m S    0  0.6 ...
20011 root      15   0 42340  22m  13m S    0  0.6 ...
23147 acue      15   0 31344  22m 4448 S    0  0.6 ...
19842 root      15   0 27016  20m 4220 S    0  0.5 ...
29259 root      18   0 43420  19m  12m S    0  0.5 ...
23249 acue      15   0  113m  16m  11m S    0  0.4 ...
2939 root      15   0 21188  16m 4836 S    0  0.4 ...
19945 root      18   0  109m  15m  11m S    0  0.4 ...
23247 acue      15   0 77020  14m 9656 S    0  0.4 ...
23264 acue      16   0 85696  13m 9.8m S    0  0.3 ...
23280 acue      15   0 85696  13m 9.8m S    0  0.3 ...
HEALTH
Total ALARMs=0
```

1

16.3 ctys Generic Options

-A <0|off|1|on>

Allow ambiguity, this has several effects on values which may or may not be allowed to be ambiguous. Allow ambiguity(-A 1) or disallow(-A 0:default) .

LABELs

Even though the labels might be ambiguous, the IDs are not, thus an unambiguous labels only restrict the access by labels, but could be used to group sessions together, if access by IDs only is sufficient.

HOSTS

When lists of hosts and groups are applied and resolved to redundant hosts within the list, this could be a desired circumstance or not. If not activated, redundancies in resulting group lists will be removed silently.

Else just executed as given.

-b <background-mode>

```
<background-mode>=
   -b
      stack
      |
      (
        (sync|off|0)|(async|on|1)
        [,
          (sequential|seq|2)
          |(parallel|par|3)
        ]
      )
```

Background and/or parallel execution. This option combines the control of detachment from console and the job distribution to multiple targets. In addition this option controls the execution of VM-Stacks, which are closely coupled to GROUPS as well as to the background mode, refer to Section 13.3.4 'Stacks as Vertical-Subgroups' on page 169 .

stack-mode
: The stack-mode is a specific enforcement of an appropriate combination for asynchronous operation sof the VMs within an sequential dependant nested VMSTACK. Therefore the values SEQ and SYNC are forced and blocked, thus could not be reset for the actual VMSTACK.

  The VMSTACK in addition decouples CONSOLE operations, though these frequently block the STDIO due to SSH only operations. The dialogue components of a VMSTACK are generally proceeded in ASYNC mode, but after the previous non-interactive task has finished. The finish of a non-interactive task is here the successful startup of a VM/PM.

detachment of jobs
: The detachment of jobs from the callers console causes the top level dispatcher to start all resulting jobs by the "-f" option of "ssh" and to return immediately. This results in a number of unmanaged jobs which implicitly are executed as autonomous parallel tasks by OpenSSH. The consequence of this is, that no higher level group functions could be performed on the whole set of results. Typical examples are

CREATE
: CREATE is performed by default as a DETACHED job, because it just creates interactive desktop sessions which are frequently not dependent on each other.

LIST
: LIST is performed by default as ATTACHED job, because it has some overall properties for tasks spanning multiple targets like SORTALL, where the individual sets could be intermixed.

parallel execution
: The high-level PARALLEL execution with ATTACHED console combines both advantages. The parallel execution reduces for bulk lists of targets the overall processing time to the slowest individual by controlled dispatch. Second it keeps the overall synchronity for performing group tasks like SORT. Typical examples are

  CREATE
  : CREATE is performed by default as a DETACHED job as stated before. In the case of DETACHED the PARALLE property has no effect, because jobs with "ssh -f" return "successfully" straight after execution.

  LIST
  : LIST is performed by default as ATTACHED job, thus the PARALLEL property has frequently a tremendous effect.
    When listing a a set of 20 machines where each requires about 15 seconds to scan all processes and calculate the results for each plugin, the overall processing time is reduced from 300seconds=5minutes to 15seconds.

Therefore the two properties complement each other, even though some similar effects could occur.

REMARK: The usage of "&" by shell expansion will just superpose the internal job control, which should avoided.

-c <args>

```
<args>=(
          ON
          [,(BOTH|LOCAL|REMOTE)]
          [,ONLY]
        )
        |OFF
```

The operations of ctys utilize special virtual-nameservice information for several reasons. These option controls the usage and selection of the caches at the local and/or remote site.

Even though for performance only aspects the usage of cached data could be suppressed, for the utilisation of stacked VMs the availability of the cache facility is mandatory.

The UnifiedSessionsManager itself is designed as a distributed client-server system operating itself by distributing it's tasks to the managed entities, and propagating required states within the vertical stack-dependencies. For this tasks at almost each point of operations ctys requires access data for it's managed objects. Therefore the UnifiedSessionsManager supports a distributed cache model consisting of multiple cache databases - cacheDBs - which has to be in sync if present, atleast might not contain inconsistent data. Partly present data will be handeled by match-priority. Alternatively the filesystem is scanned for available configuration files of VMs, once the execution target is entered.

The following flags are supported in order of control which nameservice-caches has to be used, or ignored. The default behaviour is to use "BOTH" caches, and to scan the filesystem on the execution target if no cache-hit occurs. For additional information refer to Section 9.2.1 'Distributed Nameservice - CacheDB' on page 118.

BOTH
: Use first local cacheDB, if no match occurs than delay for usage of cacheDB at the remote execution target. When no cache data is found, and the scanning of filesystem is still active, the filesystem on the final execution target is

scanned for configuration data for plugins based ob conf-files. The cache DB is created by the tool ctys-vdbgen.

This is the default behaviour.

LOCAL
: Use local cacheDB at the caller's site. The remote cacheDB on intermediate relays and on the final execution target are ignored. When no cache data is found, and the scanning of filesystem is still active, the filesystem on the final execution target is scanned for configuration data for plugins based ob conf-files. The cache DB is created by the tool ctys-vdbgen.

  For the Local option one specific point is to be considered. The basic design and resulting implementation of the locally pre-fetched cache is based on the implementation of options evaluation with permitting repetition of the same option/suboption and the "Last-Win" philosophy. Values from caches are evaluated first, before the actual user supplied entries.

  The entry available in the local cache is resolved to a fully qualified <machine-address> with all of it's actually present parts, and inserted as the first argument to the relevant ACTION of the current job from the internal schedular, before the local/remote task is finally executed. Thus user supplied parts will superpose elements from the cache and may lead to desired and/or unintended deviation from the local contents of the cacheDB. Anyhow, the iput used for evaluation of the <machine-address> from the cacheDB should be in sync.

OFF
: Deactivates both types of caches, could only used alone.

ON
: Activates cache, is set implicitly by all others.

ONLY
: Uses cache only, no dynamic data is fetched.

REMOTE
: Use remote cacheDB at the final execution target. When no cache data is found, and the scanning of filesystem is still active, the filesystem on the final execution target is

scanned for configuration data for plugins based ob conf-files. The cache DB has to be created by the tool ctys-vdbgen.

-C <args>

```
<args>=(
ON
[,KEEP]
[,ONLY]
[,RAW]
[,(FIN|FOUT):<cache-filepath>]
[,LIFETIME:<seconds>]
[,AUTO]
)
|OFF
```

For ctys two basic types of data-caches are used. The first one is the plugin specific in-mem cache, where frequent operations like "ps" for LABEL mapping will be cached for the lifetime of a process. The second is the cacheDB . on-disk cacheing, which could span multiple calls to a specific executable.
The in-mem caching is active by default, because the assumption is made that the systems state might not alter relevant to ctys within a call-cycle. This could be deactivated for the common plugins by with the "OFF" option, which additionally.

The on-disk caching is used for two specific reasons.

1. Enables collecting data for overall-processing like sort, where data from all remote tasks is prefetched on localhost into file system and postprocessed as one set.
2. Boosts performance for repetitive access to remote data, particularly when this is required within the same task in periods of seconds. Therefore an ageing timer will be set for having "neartime" data. The variable SESSION-CACHEPERIOD (default=20seconds) controls the ageing timer.

Even though the assumption that the systems state on a local machine is "more ore less static" within an uncritical ordinary call, this could not be said clearly for remote calls with on-disk caching. Thus on-disk caching of runtime data is off by default, except for collector actions, which do not reuse the "pure-data"

cache.

Following suboptions are applicable:

AUTO
: The behaviour in case of an LIFETIME exceed is changed to automatic remove of cache data. Missing files are silently created from origination. This is particularly foreseen for the internal usage of "ctys -a LIST". The AUTO suboptions is supported for local access only.

FIN:<cache-filepath>
: A user-defined cache file to be used instead of collecting remote data. This file has to contain previously cached data, which was held by usage of "KEEP" suboption. The same <cache-filepath> should be used.

FOUT:<cache-filepath>
: A user-defined cache file to be used instead of default, this is the read-write runtime cache. The filepath, if relative, is relative to "$MYTMP", but no "mkdir" is called. An absolute path is used literally.

KEEP
: Keeps cache-files instead of removing them before exit.

KEEPALL
: Keeps cache-files of all subcalls, instead of removing them before exit.

LIFETIME
: The maximum age a provided cache for "ONLY" is allowed to be. If the age exceeds, than as default the action is aborted. This behaviour is the default due to data safety.

OFF
: Deactivates both types of caches, could only used alone.

ON
: Activates cache, is set implicitly by all others.

ONLY
: Uses cache only, no dynamic data is fetched.

RAW
: Stores RAW data in cache, if not set the final results of current operation on it's actual execution-target are stored.

-d <debug-args>

```
<debug-args>=
<debug-bit-array>[,(PATTERN|P)|MIN|MAX]
[,(SUBSYSTEM|S):<subsystem-bit-array>]
[,(WARNING|W):[0-9]]
[,(INFO|I):[0-9]]
[,(FILELIST|F):<file-list>[,(EXCLUDE|INCLUDE)]]


<debug-bit-array>=2#(0|1){1,32}|[0-9]*|<any-bash-format-32bit>

<subsystem-bit-array>=2#(0|1){1,32}|[0-9]*|<any-bash-format-32-bit>

<file-list>=<file>[%<file-list>]

  DEFAULT:

  -d <#integer>

  is equal to:

  -d <#integer>,MAX,WARNING:1,INFO:1
```

Sets the level and range of debug output.

<debug-bit-array>[,(PATTERN|P)|MIN|MAX]
: The debug output could be controlled by one of two basic styles, the level-mode(MIN|MAX) or the match-mode(PATTERN).

  level-mode(MIN|MAX)
  : The level-mode sets a threshold from which on(MIN), or up to which(MAX) a trace output is displayed. The switch-on value has to be increment one above the destination output level.

  match-mode(PATTERN)
  : The match mode displays trace only by bitwise AND operation.

  The debug mode value could be provided in any bash supported notation, but only 32bit arrays should be used.

  ```
  <debug-bit-array>=2#(0|1){1,32}|[0-9]*|<any-bash-format-32bit>
  ```

  The following variables are predefined to be used for levels and pattern.

ERRORS
Traced independently and in any case.
D_UI=1=2#1 Common UserInterface.

D_FLOW=2=2#10
Common UserInterfaceExtended, call flow.
D_UID=4=2#100
Common UserInterfaceDebug, draft data collection.
D_DATA=8=2#1000
Detailed data processing.
D_MAINT=16=2#10000=16#10
Maintenance, details of attribute evaluation.
D_FRAME=32=2#100000=16#20
Traces the framework.
D_SYS=64=2#1000000=16#40
Traces system calls encapsulated by "callErrOutWrapper". Particularly useful for evaluating the required root-permissions for "ksu" and/or "sudo".
D_TST=16384=2#100000000000000=16#4000
Traces sync-points for regression tests.
D_BULK=32768=2#1000000000000000=16#8000
This is the the haystack.

(SUBSYSTEM|S):<subsystem-bit-array>
Subsystems as match-mode bitr array.

```
<subsystem-bit-array>=2#(0|1){1,32}|[0-9]*|<any-bash-format-32-bit>
```

The following variables are predefined to be used for subsystems.

S_CONF=1
S_BIN=2
S_LIB=4
S_CORE=8
S_GEN=16
S_CLI=32
S_X11=64
S_VNC=128
S_QEMU=256
S_VMW=512
S_XEN=1024
S_PM=2048

Common generic values:
Values to be used for multiple categories.

D_ALL=65535=16#ffff
This activates all.

(WARNING|W):[0-9]
Warnings to be displayed, level-mode only and no subsystem. "0" switches off. The switch-on value has to be increment one above the destination output level.

(INFO|I):[0-9]
Info to be displayed, level-mode only and no subsystem. "0" switches off. The switch-on value has to be increment one above the destination output level.

(FILELIST|F):<file-list>[,(EXCLUDE|INCLUDE)]
A list of files to be included exclusively or excluded. The names are matched with the presented string on output "<dir>/<file>", where due to performance reasons a simple pattern-match is performed only. For the same reason the EXCLUDE and INCLUDE keywords are applied to the whole set at once.

```
<file-list>=<file>[%<file-list>]
```

-D <DesktopId>|<DesktopLabel>
This parameter requires the tool 'wmctrl' to be present, if not the usage is not provided and an error message is generated before exiting ctys.

When provided by system and successully detected, the following applies:

<DesktopId>
The id of the desktop to be used for placing the window. Currently 1-3 digits are supported.

<DesktopLabel>
The user defined label of the desktop to be used for placing the window. When beginning with a digit, and is shorter than 4 characters, at least one character has to be a non-digit, otherwise it will be detected as <DesktopId>.

Currently special characters like '&' are not supported, so just digits and ordinary characters and hyphens should be used.

A list of current desktops could be shown by calling:

"wmctrl -d"

Where the first column is the id of the desktop, and the last is the label. For further information on wmctrl refer to related man page.

In current implementation of ctys two cases for the addressing of desktops has to be distinguished.

- Display Forwarding In case of Display Forwarding the window is not managed by target Xserver, just it's window is drawn. So this window could not addressed by wmctrl, and it is not listed when using 'wmctrl -l'. So this case will be handled by switching the desktop to the target, and semi-sysnchronous starting the session, until the window appears on the desktop.
- Connection Forwarding In this case the client is executed on the local machine and is - if supported - managed by the target Xserver. But, this seems not to be reliable.

So in any case, when the desktop is specified, ctys switches to synchronous operations and switches one after another to all desktops where a window has to be depicted. Current processed desktop is the visible desktop, until any assigned window has appeared.

LIMITS:

Due to restrictions of current wmctrl versions window-selection the sessions for each destop are called grouped together and placed on the defined desktop by previous switching of visible desktop before calling configured sessions.

OK, so far so good, BUT when using SSH and requiring authentication by password - which is when no SSO is implemented - due to call convention of SSH(ssh -f ...) the password is requested in the callers CLI/XTerm. When the selected output desktop is different from that, it will be switched before calling SSH-session of course. This means it leads to a now hidden password-request on the calling desktop.

There is - due to basic SSH concept - NO SCRIPTING WORKAROUND to this behaviour when no reliable "moving windows between desktops" support is available. So the possible solutions for now are:

1. Establish Authentication on SSH level, by key-distribution or usage of kerberized SSH.
   Obvious, but once again: Please not by rhost, and not for root at all. Probably you can integrate ksu into your login. Avoid remote access permissions for root in any case!
2. Make the calling terminal visible on all your desktops, at least during ctys configuration call.

-e

For completeness only: Execute locally, this indicates the termination point of a virtual circuit. This parameter is foreseen for internal usage only, and must not be used, by ordinary users, thus it is not further detailed.

-f

Force execution and ignore minor warnings. Basically no "destructive" operation, particularly nothing irreversible will be performed.

-F <remote_version>

Force remote version.

-g <geometry>|<geometryExtended>

The geometry for client-side representation. It is the exact syntax of X client "–geometry" parameter with an additional screen parameter as alias or index for usage with Xorg multiple displays.

ATTENTION: In order of using xorg.conf and saving effort some minor assumptions as requirements concerning the xorg.conf file are made. Current implementation requires due to stateless filtering the field "Identifier" as first entry in "ServerLayout" sections.

Supported variants:

Xorg-style: <geometry>
"<x-size>x<y-size>[[+,-]<x-offset>[+,-]<y-offset>]"
Any screen offset has to be calculated manually.

Xinerama-alias-style: <geometryExtended>
"<Xorg-style>[:[<ScreenSection>|<ScreenIndex>][:<ServerLayout>]]"

"<x-size>x<y-size>:Screen4"
The screen from the first ServerLayout section with given Screen section name as alias will be used. The required offsets will be calculated from the "/etc/X11/xorg.conf" file.

"<x-size>x<y-size>:Screen4:Layout[0,1]"
The screen from the LayoutSection named "Layout[0,1]" with given Screen section name as alias will be used. The required offsets will be calculated from the "/etc/X11/xorg.conf" file.

"<x-size>x<y-size>:4:Layout[0,1]"
The screen from the LayoutSection named "Layout[0,1]" with given Screen index will be used. The required offsets will be calculated from the "/etc/X11/xorg.conf" file.

-h

ATTENTION:
Refer to "-H" for additional information.

Help, Shows a short sum-up of call convention and possible call details for getting additional help.

-H <item[,item [,item]...>]

ATTENTION:
Due to the amount of documentatin required, which is currently already more than 400 pages, the "inline documentation" is shifted to a LaTex document and is provided as a complete document in one of the formats:

- PDF - online, with extended table of contents for call-Hyperlinked bookmarks.
- PDF - offline, whith some reduction for paper prints.
- HTML - generated by texi2html.

Therefore parts of the online help from the commandline are temporarily unavailable, and under construction in general.

As soon as a generic approach is decided to produce ASC-II help text from LaTex-sources this will be included.

The "-H" option shows detailed description and extended help. Therefore it is possible to constrain the output by choosing a list of specific options or any other item literals from the help text. For displaying of the whole text any of following generic options could be used:

"ctys -H ”"

"ctys -H all"

"ctys -H ’*’"

For displaying of the whole text with formatted output for printing the following option could be used. The same text as with "-H all" is printed than, but with filtering and formatting by "pr". The variable "PR_OPTS=${PR_OPTS}" could be pre-set. Current default is "-o 5 -l 76" for DIN-A5.

"ctys -H print"

For displaying a specific subset of the help text an literal item could be specified. The given literal will be matched at the beginning of leading non-whitespace and non-commentary lines within the output of "ctys -H all". So e.g. to find help on how to list functions just type "ctys -H func". But be aware, the help content may change, therefore this searched entity could be obsoleted in a later version. For listing the description of option ’-H’ only the following has to be called:

"ctys -H ’-H’"

For listing the description of both options ’-H’ and ’-h’ only the following has to be called:

"ctys -H ’-h,-H’"

So for listing the help only for any list of options (with hyphen) just type the list of suboptions as for ’-h,-H’. For listing of additional literals from text now a short description of the item literal is neccessary.

SECTION-KEYS

Each given item is used internally as a keyword in order to determine the beginning of a section with it's own description. Multiple sections for an item are supported. The decision when an item is matched, or it's description section is starting and finishing is made based on indention of the text. Therefore the starting point is searched as a string given by the following regexp

"*"<item>

or

precisely by '$1 "^"_a' in awk, where _a is the value of item, where the seperator is default.

Even though leading white-spaces in the item - if less then the search target - will function well, they are eliminated by intermediary shell-expansion within the processing script. Thus keys are only supported without spaces.

This means, that the first string is matched, which consists of non-whitespace characters and is terminated by a whitespace. The matching line is the first line of description.

Next the indention is utilized in order to decide the termination of the specific description. Therefore the leading white-spaces are used to assemble a new regexpr in order to search for the first terminating line of current description, which is excluded.

The item could be a shortcut of the targeted keyword, but should be unambiguous, otherwise any matching item - which is some more - is listed.

The only one exception to this is that leading "=" are ignored, this is in order to underline visually main headers - beeing could-be items of choice - with equivalent width to the search item itself.

Another important fact is, the item is evaluated literally, which is case sensitive of course.

For listing the description of "SECTION-KEY" only type:

"ctys -H SECTION-KEYS"

So for now the description might be sufficient to understand the options for items to be displayed. For example the "DESCRIPTION" section will be displayed with one of the following calls:

"ctys -H DESC"

"ctys -H DESCR"

"ctys -H DESCRIPTION"

But nothing is displayed with the following calls:

"ctys -H DESc"

"ctys -H DESCRIPTIOn"

"ctys -H DESCRIPTIONx"

Additional following keywords are supported for development interfaces. Literal bash functions and Script header are distinguished by TYPE(bash-function vs. bash-script).

funcList[=[<any-function>][@<module-name>[@...]]]
: List of function names, sorted by function names. In addition the file names and line numbers are displayed too.

funcListMod[=[<any-function>][@<module-name>[@...]]]
: List of function names, sorted by file names. In addition the file names and line numbers are displayed too.

funcHead[=[<any-function>][@<module-name>[@...]]]
: List of function headers, sorted by file names.

The following constraints could be applied:

<any-function>
: If given <any-function> than only this is displayed.

<module-name>
: If given <module-name>, than the functions contained within this module only are displayed.

-j <job-id>
: The ID of current job. This is an internal call and therefore should just used by developers for test purposes. Any variation of the JOB_IDX for the CLI call may severe job execution seriously and even can damage user data when set for a CANCEL operation. The originating CLI call should not use this option in productive operations, any subcall may have a propagated value as required. The value within the originating interactive CLI call is set to "JOB_IDX=0". This is the value

for the starting point of internal task-scheduler data, and thus the index for the first performed task too. The value is evaluated by plugins for handling job specific PROLOG and within EPILOG for decision of the state of passed job and eventual required post-processing.

In addition a variable "CTYS_SUBCALL" is set.

"-j ${DATETIME}:${JOB_SUPER}"

default:"CALLERDATETIME=${DATETIME}"
"CALLERJOB_IDX=${JOB_IDX}"
"CTYS_SUBCALL=${1}"

-l <login-name>

The users, which will be used for hosts without an exlicitly given user. The hosts/groups entries provide the common EMail-Style "<user>@<execution-target>". The default is "$USER", when neither "-l", nor an explicit user is provided.

-L <execution-location>

```
<execution-location>=(
(LOCALONLY|LO)
|(CONNECTIONFORWARDING|CF)
|(DISPLAYFORWARDING|DF)
|(DisplayRedirection|DR):<display>[.<screen>]
|(CLIENTONLY|CO)
|(SERVERONLY|SO)
)
```

This option controls the location and possible split of the involved client and server parts of current session. When connecting a user interface with it's server components the following basic constellations could be distinguished:

LOCALONLY
Client and server components are coallocated on users workstation, the display is driven locally.

DISPLAYFORWARDING
Client and server components are coallocated on server, and the display is forwarded to the users workstation.

DisplayRedirection:<display>.<screen>
Almost the same case as DISPLAYFORWARDING, the

only difference is that a redirection of the display on "localhost" will be done. The localhost is the only supported redirection, due to security issues.

CONNECTIONFORWARDING
Client and server are split, whereas the client component is located on the users workstation and the server component is located on the server machine. The connection from server/client component is forwarded on application protocol level.

CLIENTONLY
This is a client in standalone mode, still used internal only, e.g. for COPY when performing phase-2 with remote-copy from local client.

SERVERONLY
This is a server in a so called headless-mode.

-M <message>
Free text to be used as prefix for target exec. It will be printed before output.

default:""

-n
Just display, do not execute. For test only.

-p <db-directory-path-list>
Path list to directories containing DBs for name resolution, same for each <db-directory-path> as for ctys-vdbgen. ctys will internally handle names by multiple levels of resolution, which depends on the actual executing plugin. The most sophisticated address resolution is frequently required for VMs when using them in a roaming manner on groups of machines, where after some plugin specific resolution of convenient VM-addressing by user an TCP/IP service-access-point for OpenSSH has to be addressed.

The second case to be handled is the addressing of execution entities of type HOSTs transparently within a VM or PM. This will be supplied by ctys-nameservice too. For almost all of the nameservice tasks additionally required for plugin-specific address resolution actions the ctys-vhost command is used internally. This option sets the databases for operations of ctys-vhost. If not present ctys-vhost defaults will be applied.

-P

Use default <db-directory-path>.

-r <xsize>x<ysize>

Remote resolution, which is by default the same as local client size given by "-g" option. This configures the virtual graphic card of the server with the provided resolution. This parameter is not applicable to any application. It has to be defined in the application specific package. The current supported applications are:

VNC: for vncserver

-s <scope>

Restrics/expands the scope/selected set for mode of operations.

USER
Own sessions.

GROUP
Sessions of own group.

USRLST=(usr1[,usr2][,...])|all
Given list.

GRPLST=(grp1[,grp2][,...])|all
Given list.

ALL
This value is remapped to "USRLST=all".

-S [(on|off)][,][<ignore-signal-spec>]

Sets the signal spectrum to be ignored. The values are accepted as numeric values only. Applicable values could be displayed by "trap -l"(within bash).

The default values are "1,3,19", which is set for CLI0 consoles only by default.

In case of CLI, generally for any multi-session call, it has to be considered thoroughly whether and which signals could be set.

-t <session-type>[=suboptions]

Defines the context of execution and the resulting applicable feature set. This could be a flat endpoint-user-session in case of VNC, or a virtual OS-starter in case of a VM session e.g.

in case of VMW or XEN. Suboptions specify more detailed characteristics. Thus this parameter has to be set first.

<session_type>(default:VNC)

VNC

Remote VNC sessions, calls the scripts ctys-callVncserver and/or ctys-callVncviewer. The specific behaviour is here to set a password for the VNC session from a passwd-file via CLI option. The access rights of this stored passwd in

$HOME/.vnc/passwd

should be checked.

For additional types refer to the specific plugins.

To load multiple plugins for one call, the environment variable CTYS_MULTITYPE or the "-T" option could be set.

-T <session-type>[,<session-type>[,...] | all]

Preloads given list of <session_type> instead of loading the plugins of requested types by "-t" option. Alternatively the environment variable CTYS_MULTITYPE could be pre-set, which has the same result. If CTYS_MULTITYPE and the "-T" option are provided, the option has priority.

This option is required for the scope control of generic actions, which generally will be applied by calling of all current loaded <session_type> interfaces. E.g. the "-a LIST" action lists active sessions for all actually loaded <session_type>. For display of current active sessions of all available <session_type>, the "-T all" has to be used.

<session_type>

The name of a dynamic loaded plugin, which is the <session_type>.

all

Tries to load all present plugins, this would frequently fail, when the configured resources of bash are exceeded.

This could be even caused by a single module, which exhausts available resources - as in any existing system.

Thus the default will be set to requested types by "-t" options or NIL by default.

-v

Show version. Current version scheme is as follows:

-V

Show version. Current version scheme is as follows:

AA_BB_CCC[abc]DD

AA:
Official major upgrades.

BB:
Official minor upgrades.

CCC:
Build <-> Test versions.

abc :
Development versions, Test-States:

a lpha

b eta

c (g)amma

DD:
Pre-Release development versions. Anyway, if publicly available might be yet almost stable.

This option strongly interacts with the "-X" option, when set only the version number is display, without a <CR>. This is the only relevant information for batch-processing. Else all current loaded components - libraries + CORE-plugins + Application-plugins - are listed.

Using this option twice shows in addition to the plugin short-names the actual file of storage. Sub-packages loaded by Application are contained in the list too.

This list is generated at the end of execution, thus on-demand-loaded sub-packages are listed too, as far as the have been demanded during current call. The set of the "on-demand-loaded" plugins can vary in dependency of the actual performed control flow.

When using this option alone, only the initial by "default-loaded-components are listed.

-X

Generate terse output for post processing. The '-v Verbose' flag is not effected and should be only used for testing.

-y

Activates some terminal capabilities, mainly coloring of ERROR, WARNING, and WARNINGEXT. Very handy when debugging, but not yet supported for Emacs-Consoles. As an alternate the variable "CTYS_TERM_COLORS" could be set to "0". When selected the local and remote settings are both set at once. In current version this is set by default when the variable TERM is set to "xterm".

-z (NOPTY|PTY|1|2)

Control the allocation of a pseudotty by ssh. Therefore one or two "-t" options could be set for the internal "ssh" call.

```
NOPTY    : Eliminates "-l" of standard bash-call
           and "-t" for ssh-call.
PTY      : "-t"
PTY,PTY  : "-t -t"
1        : "-t"
2        : "-t -t"
```

-Z (KSU|NOKSU|SUDO|NOSUDO|ALL)

Controls call permission-grant. The calls requiring impersonation to another users ID, frequently "root" for restricted system resources, are supported to use "ksu" and/or "sudo". This option replaces the default settings from the configuration file. The mechanisms could be switched on/off selectively.

```
KSU      : use Kerberos
NOKSU    : do not use Kerberos (DEFAULT)
SUDO     : use sudo
NOSUDO   : do not use sudo     (DEFAULT)
ALL      : use al provided
```

As an persistent alternative following environment variables could be pre-set.

```
USE_KSU (0=>off 1=>on)          (DEFAULT:0=>off)
USE_SUDO (0=>off 1=>on)          (DEFAULT:0=>off)
```

The evaluation is implemented as a generic check for first match of hard-coded call-check. The following order of permission tests is performed for each system callee.

1. user is root
2. native access granted
3. ksu call
4. sudo call

In case of 2.) the current ID is checked for "$USER==root", if not, than a warning is generated, but continued with procedure. This is due to possible security flaws, when assigning root-ID to an ordinary user. Anyhow, when using ctys from a system account, this might be OK.

REMARK:

When a user cannot be authentified by one of sudo or ksu, then the system waits for a user interaction. BUT, due to internal "silent" checks the stdio was redirected, of course. Thus the system seems to be "hanging", or requests "Password:" and seem not to continue afterwards, but it "may work".

This is a "natural dilemma", because within the generic check function called for each task several times the output has to be suppressed. Currently no detection for an exceptional User-Dialog request is implemented. So for now are two diagnosis facilities implemented:

1. A warning as "**HINT**" is generated, when "-w" option for extended warning is set. This shows the wrapped native call, which should be called manually by cut/paste on the ACTUAL EXECUTING system.
2. The system variable CTYS_NOCALLWRAPPER could be set, which deactivates the wrapping of stdio and stderr for the call wrapper only.
   This means, that some redirection for the call context is still active, because it is a required output data, or is simply bulk data which might flaw the whole sense of diagnosis.

When typing a RETURN the process will continue, but disabling the current type of permission mechanism. This could be OK, when KSU and SUDO is set, and KSU has no permissions configured, but SUDO has. It could lead

to an later error too, when none could be detected. This scenario occurs for:

1. KSU: Kerberos credential was timed-out.
2. SUDO: User and/or call are not configured in sudo for execution target.
3. On client machine no permissions for system calls are configured. This case can frequently be ignored safely.
4. The only case where this can lead to an error is the missing permission for access to a proprietary client application.

When using "sudo" the flag "requirestty" within "/etc/sudoers" control whether a TTY is required or not. When in order to avoid this uncomment the flag within sudoers file. The "-z" option could be used to activate a pseudotty.

16.4 ctys Arguments

```
-a <action>[=<action-suboptions>]
[--]                                          \
['('<any-options global for all remote>')']  \
(([user@]<hostname>)|<groupname>)['('<any-options>')'][ ...]
```

These are the remote options which are given as global and individual options for each host. The options are (almost) the same as for common call.

ATTENTION: .

"–" is required when using remote options, otherwise some problems with standard remote options might occur.

<user>

default:$USER=$USER

When instead the "-l" option likewise the r-functions is supported the given user(list) is permutated with the listed hosts. Particularly nice for bulk-tests, but anyhow a limit of about 20 sessions to individual hosts (IP-addresses) is hardcoded to avoid some "hard-coded-resets". This value could be reset by following environment variable:

R_CREATE_MAX=$R_CREATE_MAX:-20

<hostname>
default:'uname -n'(‘uname -n‘)

<groupname>
Any user defined group/macro, for additional information refer to ctys manual.

<any-options>
Any global option could be provided individually for each host. E.g. individual debugging level on that host only.

One implementation specific to be aware of is, that these options are superposed, but not reset, thus the current environment will remain for the following host.

The following example shows three hosts, where each has a different debugging level.

First of all the debugging flag and level is not forward propagated, and as common for all other environment settings too, "the last wins".

```
"...-d 6 -- ( -d 3 ) host01 host02( -d 1) host02(-d 0)..."
```

So, the given options results in the following scenario:

```
-> localhost:   -d 6          = -d 6
-> host01:      -d 3          = -d 3
-> host02:      -d 3 -> -d 1  = -d 1
-> host03:      -d 3 -> -d 0  = -d 0
```

<command>
Command to be executed on the target host, which could be a native physical providing a remote desktop based on VNC, or a virtual machine like Xen. In any case the login will be performed by means of the target system, but the administrative support for neatless execution is provided by this tool.

16.5 ctys-plugins

Chapter 17

Plugins - Feature Extensions

This section contains help for the add-on plugins contained in the distribution of ctys.

These are designed to be executed on "PM" which are PhysicalMachines in addition to "VM" which are VirtualMachines.

17.1 Prerequisites

Following pre-requisites have to be prepared for the listed plugins. The specific plugins may have additional requirements, such as installation of a specific GuestOS and usage of specific drivers. These are listed for the plugins specifically.

	VDE TAP	sudo ksu	kernel	HOST OS
CLI				L,B
X11				L,B
VNC			(3)	L,B
QEMU	X		(1)	L,(4)
VMW			X	L
XEN		X	X	L,(5)
PM			(2)	L,B

Table 17.1: Prerequisites and applicability of plugins

(1) KQEMU is currently not supported within stacks, could be used only for the "lowest" VM of a stack.

(2) Depends on upper stack, for VMs almost in any case required.

```
(3) RealVNC is tested more thorougly, but Tight VNC is supported
    as well.

(4) Currently Linux only is tested, but following will be
    supported soon:

     - FreeBSD, NetBSD, OpenBSD
     - OpenSolaris

(4) Currently Linux only is tested, but OpenBSD, FreeBSD, and
    OpenSolaris will follow.

(5) Currently Linux only is tested, but following will be
    supported soon:

     - NetBSD (OpenBSD, and FreeBSD as soon as available)
     - OpenSolaris


 L  Linux: The preferred distribution is CentOS-5.x.

 B  BSD:   Currently OpenBSD only, but FreeBSD and NetBSD will follow.
```

A common requirement - as already mentioned - is to establish a distributed Authentication and Authorization system, e.g. SSO. Otherwise a recognizable amount of password demands might occur.

Any supported GuestOS of listed VMs will be supported. For non-listed HOST-OSs used as GUestOS no native access based on CLI,X11, or VNC exists, so for CREATE and CANCEL the hypervisor could be used only. A controlled shutdown is particularly only possible by means of the controlling hypervisor. So no stack-propagation will be performed.

Some effort is foreseen to be spent for the integration of the OSs FreeBSD, NetBSD, and OpenSolaris, and missing features of OpenBSD.

The migration to other UNIXes is open to demand, and depends on vendor's support by systems donations.

Additional proprietary OSs might be probably handled too by usage of Cygwin within a future version.

Some GUI effort might be spent, but is beeing honestly not really within focus for now, same as for Cygwin support.

17.2 Category HOSTs

This section contains the add-on plugins of category HOSTs, which are executed native on OS level for native interaction with hosting GuestOSs as well as for controll sessions to the containing OS of the targeted hypervisor.

These are the standard remote sessions based on CLI/Console, X11, and VNC, the workhorses of the UnifiedSessionsManager for interactive user sessions as well as for remote execution of batch commands.

17.2.1 CLI - Command Line Interface

Description

The CLI-plugin starts simply a remote shell from within a local shell. No specific default desktop functionality such as XTerm is supported, just a pure CLI access is performed. The handling of own desktop windows such as XTerm or GTerm will be supported by the X11-Plugin.

The execution of the interconnecting ssh-call will be performed by setting the "-X" option for automatic X11-Portforwarding. Thus even though the CLI session itself is not GUI based, any X11 XClient could be started and will be displayed on the local caller's XServer by usage of the encrypted channel. Thus an Xterm or gnome-terminal could be started from within a CLI sub-option "CMD", refer to the chapter "Examples".

Chained login by gateways with Overall-Display-Forwarding is supported by OpenSSH, thus by ctys too. This is for example required for the PM plugin, when an intermediate relay with root-permissions is required for sending a WoL packet.

When executing CLI almost the same functionality as for an ordinary SSH session is supported. The main advance of using ctys instead of an ordinary ssh-call is the seamless integration of native ctys-addressing for execution targets. Resulting from this feature instead of an IP address the <machine-address> can be used. Thus the administration of address-to-target-mapping could be completely delegated to ctys and will be utilized by internal usage of the full scope of ctys-vhost. E.g. the LABELS defined within ctys could be used to open a remote shell to any "labeled" instance such as a

VM or PM. Therefore internal CLI type CONSOLES only use the CLI plugin.

Another option might be the usage of UUIDs or MAC-Addresses for persistent definition of the handling of commands on frequently changing and reinstalled systems with reassigned IP-Addresses and DNS names. This could be extremely valuable for extensive tests on large systems with different scenarios and participating member collections.

The second important feature of CLI is the facility to start native remote commands as shell calls within any managed PM or VM instance. Therefore CLI is the working horse for native execution of GUI-less ctys-tasks.

Similar to the call of a system command, any internal ctys library and plugins function could be called remotely by CLI sub-option "CMD". Refer to examples to "procFindTopBottom" in the chapter "Examples".

Due to the simple remote-shell nature of CLI, just a few of the "Client/Server" based actions are applicable. For example a CANCEL is currently not foreseen to be supported, because the exit from the interactive call or started command closes the line by default. Asynchronous user specific commands has to be managed by the application and/or user.

The default shell used on the target instance is bash, which could be altered interactive by the sub-option "SHELL|S", or persistently by the variable
CLI_SHELL_CMD_DEFAULT.

The basic call behaviour is controlled by the common ctys options. The CLI operates by default as a synchronous sequential foreground process, handling all <execution-targets> one by one. This behaviour could be altered by the "-b" option to asynchronous background mode, and/or to parallel distribution. The usage of a PTY is controlled by the "-z" option.

ATTENTION: The ONLY and ONE thoroughly tested shell is the bash.

Options

-a action[=<suboptions>]

CANCEL
Not applicable.

CREATE

```
CREATE=[<machine-address>]
    [REUSE|CONNECT|RECONNECT|RESUME]
    [CONSOLE:<>]
    [(CALLOPTS|C):<callopts>]
    [(XOPTS|X):<xopts>]
    [(SHELL|S):<shell>]
    [(CMD):<cmd>]
```

<machine-address>
See standard for syntax. Due to the limited attribute set of a completely dynamic CLI session without own persistent resources only the LABEL is applicable. The hosting OS has to be handled by it's own plugin. The LABEL sub-option is here mandatory.

<callopts>
Refer to common options description .

CMD:<cmd>
Refer to common options description .

CONSOLE
Not yet supported.

<xopts>
Refer to common options description .

REUSE|CONNECT|RECONNECT|RESUME
Not applicable.

(SHELL|S):<shell>
Refer to common options description .

A specific exception to the synchronous character of a CLI shell occurs, when multiple execution instances are addressed by the arguments of current call. When the "-b off" option for background a.k.a asynchronous operations is selected, the standard synchronous foreground operation

works quite well. The list of arguments is just executed sequentially as expected. The CLI CREATE action sets implicitly the forced usage of a pseudotty. This is the same as using "-z pty,pty".

ENUMERATE
Not applicable to CLI.

LIST
Almost the same output as common standard, with following changes in semantics.

id The PID of the local SSH termination point, which is the locally executed relay-instance of ctys and is the parent shell of actual running batch/interactive shell. The CLI plugin does not support a cancel or connect|reconnect|reuse action, so the "id" is here a non-functional hint only.

pid PID of current ctys sessions top.

The following values are not applicable:

uuid, mac, dsp, cp, sp

-L (LOCALONLY|LO) | (DISPLAYFORWARDING|DF)
Even though a DISPLAY will actually not be used, this parameter also adopts the behavior of the execution to a providing environment, which is frequently for ctys true.

These are the only location parameters to be applied.

Examples

Refer to Section 21.1 ‘CLI Examples’ on page 429 .

17.2.2 X11 Interface

Description

The X11-plugin starts a remote shell within a terminal emulation by default. Alternatively any X11 command with an arbitrary shell could be executed. When executing a command, the functionality is almost equivalent to the CLI plugin.

The primary application of X11 is to utilize the default behaviour of starting an Xsession within it's on terminal emulation on the desktop. Therefore the pre-configured choices are: XTERM, GTERM, and EMACS, which is started in "shell-mode".

Options

-a action[=<suboptions>]

CANCEL
Not applicable to X11.

CREATE

```
CREATE=<machine-address>[,]
   [(REUSE|CONNECT|RECONNECT|RESUME)][,]
   [
    (CONSOLE:XTERM|GTERM|EMACS|EMACSA)
    |
    (CMD:<cmd>)
   ][,]
   [(CALLOPTS|C):<callopts>][,]
   [(XOPTS|X):<xopts>][,]
   [(SHELL|S):<shell>][,]
   [(DH|SH)][,]
   [(TITLEKEY:<key-name-label>|NOTITLE)]
```

<machine-address>
The LABEL suboption is here the only supported and mandatory part.

<callopts>
Refer to common options parts description.

CMD:<cmd>
: Refer to common options parts description. The CMD could be provided alternatively to a fixed CONSOLE.

CONSOLE:(XTERM|GTERM|EMACS|EMACSA)
: A fixed type of a CONSOLE, folowing types are supported.

  CONSOLE:GTERM
  : Starts the "gnome-terminal".

  CONSOLE:XTERM
  : Starts the Xterm.

  CONSOLE:EMACS
  : Starts an emacs and opens a "shell" buffer.

  CONSOLE:EMACSA
  : Starts an emacs and opens an "ansi-term" buffer.

(DH|SH)
: Sets a double-hyphen or a single-hyphen for call arguments. Where the single hyphen e.g. is required for old style X11 utility options, the double-hyphen for new options style of Gnome.

NOTITLE
: Suppresses the generation of title either from LABEL, or from default.

  ATTENTION
  : Be aware, that dynamic sessions like X11 without an LABEL visible by ps, are not - or just limited - recognized by LIST action. Thus the only reliable support for sessions started with this flag is the start itself.

REUSE|CONNECT|RECONNECT|RESUME
: Not applicable.

(SHELL|S):<shell>
: Refer to common options parts description.

TITLEKEY:<title-name-key>
: Alters the options keyword to be used to set the title of an Xwindow, which is set by default to "title". Some older applications, like Xclock support different, e.g. "name" as options keyword only. Due to the crucial role of the window title, which is the LABEL, this approach should be prefered when "title" is not supported. Do use NOTITLE only if definetly unavoidable.

<xopts>
Refer to common options parts description.

A specific exeception to the sysnchronous character of an CLI shell occurs, when multiple execution instaces are addressed by the arguments of current call. When the "-b off" option for background a.k.a asynchronous operations is selected, the standard synchronous foreground operation works quite well. The list of arguments is just executed sequentially as expected.

In current implementation the user is responsible for handling the appropriate values, which are assigned by default. The enforcement of resetting user defined values could be somewhat tricky due to permutation of bulk arguments, thus is shifted because of priorities. Resulting of this, the actual environment is a superposition of all previous executed target options with the global options.

ENUMERATE
Not applicable to CLI.

LIST
Almost the same output as common standard, with following changes in semantics.

[id]
The PID of the first local call below of SSH termination point, which is the locally executed relay-instance of ctys and is the parent shell of actual running batch/interactive shell. Thus the topmost ctys-call.

[pid]
PID of current ctys sessions top.

The following values are not applicable:

uuid, mac, dsp, cp, sp

-L (LOCALONLY|LO) | (DISPLAYFORWARDING|DF)
Even though a DISPLAY will actually not be used, this parameter also adopts the behavior of the execution to a providing environment, which is frequently for ctys true. So these are the only location parameters to be applied.

Examples

Refer to Section 21.2 'X11 Examples' on page 437 .

17.2.3 VNC - Virtual Network Console Interface

Description

This package just starts a VNC session on the native OS on the target target host. Therefore a vncserver is started and managed on the target server, whereas a vncviewer could be started on the target host or a any client by "Display Forwarding" or "Connection Forwarding". The intermixed usage of VNC is supported too, where the vncviewer is connected to a VM, is handled within the specific VM-package. This is for example the case for Xen. Currently the VNC variants RealVNC (main target) and TightVNC are supported.

Sessions Name binding and Bulk Access

The standard namebinding is extended for the VNC plugin with the attribute BULK. This is particularly handy for testing purposes, but helpful for a number of daily tasks too. The opportunity of openning multiple sessions at once is particularly useful for DISPLAYFORWARDING, because in this case all sessions are forwarded through one shared SSH tunnnel. This is initiated by the BULK evaluation and disassembly of the action to multiple calls on servers site. Thus the X-forwarding of OpenSSH bundles all local executed XClients within one tunnel.

When using CONNECTIONFORWARDING, business is as usual, multiple tunnels are created, one for each client. This is required, because the "ControlMaster" feature of OpenSSH does not support X-forwarding by different displays in current version.

The second specific, as for all dynamic allocated system IDs, is that the IDs are not applicabe to CREATE action, here just a mandatory LABEL is provided. The resulting applicable <machine-address> is as following:

```
<machine-address>=(
(id:<id>)
|
((label|l):<label>)
|
(bulk:[0-9]{1,3},(label|l):<label>)
)
```

REMARK:

In current version the CREATE action frequently fails to attach a viewer to a started server ONCE only, when started

for the first time. A following CONNECT or REUSE will attach a vncviewer as expected. This happens due to some X-permissions error on NFS mounted home directories, for specific faster machines only.

> "X Error of failed request: BadAccess (attempt to access private resource denied)"

This is a planned error correction/workaround for one of the next versions.

Options

-a action[=<suboptions>]

CANCEL

```
CANCEL=(<machine-address>)\{1,n\}
  |ALL
  (
    [FORCE|STACK][,]
    [SELF][,]
    [
      RESET|REBOOT
      |(INIT:<init-state>)
      |(PAUSE|S3)|(SUSPEND|S4)
      |((POWEROFF|S5)[:<timeoutBeforeKillVM>]
    ][,]
  [CLIENT|SERVER|BOTH]
  )
```

<machine-address>

For VNC the following parts of a <machine-address> are applicable:

> ID|I, LABEL|L

When the VNCviewer/VNCserver is used in shared-mode, which is the default for ctys, the address applies to all sharing VNCclients/vncviewer are handled as one logical unit and CANCEL is applied to all at once.

The address could be supported with multiple instances.

ALL|BOTH|(CLIENT|SERVER)
: These define the range, where ALL and BOTH kill clients and servers on local machine. Remote clients by CONNECTIONFORWARDING might be exiting when server-loss is detected.

  The SERVER scope is actually for VNC the same as ALL or BOTH, this is due to the default (non-server) behaviour of attached clients, which exit when detecting a server-loss.

  The CLIENT scope just kills all client processes by means of OS, which is simply calling kill on their PID. The server processes remain untouched.

REBOOT|RESET|INIT|SUSPEND
: These methods just behave as a "soft-kill" which is a more or less soft shutdown, for VNC only! Application shutdown is not supported. So in this case first all clients are killed, following a call to "vncserver -kill :<id>" for all matched. No additional action is performed in case of a failure.

POWEROFF
: These method could be seen as a "hard-kill" which is a trial to "soft-kill" and an immediate following process kill by means of native OS. Thus there might be definetly no difference to a controlled shutdown of VNC managing unprepared applications.

The session(s) are basically just killed, so the caller is resposible for appropriate handling of contained jobs.

CREATE

```
CREATE=[<machine-address>]
  [,(REUSE|CONNECT|RECONNECT|RESUME)]
  [,BULK:[0-9]{1,3}]
  [,VNCBASE:<base-port>]
  [,VNCPORT:<literal-port>]
  [,WAITS:<delay-before-viewer-call>]
```

<machine-address>
: For VNC the following parts of a <machine-address> are applicable:

  LABEL|L

When the VNCviewer/VNCserver is used in shared-mode, the address applies to all sharing VNCclients/vncviewers. The LABEL suboption is here mandatory.

BOOTMODE
Not applicable.

CONNECT
Almost the same as REUSE, but no new server will be started if missing.

CONSOLE
Not applicable.

PING
Not applicable.

RECONNECT
Similiar to REUSE, with the difference, that any previous active client will be killed before attaching ONE new client. Therefore in shared mode, when multiple clients could simultaneously share one server, all sharing clients are handled as one logical unit and will be thus killed together. Specific exchange of a single client is not supported.

RESUME
Not applicable.

REUSE
When a server process with matching ID or LABEL is already running it will be used, else a new one will be started. In case of non-shared-mode operations of VNC any running vncviewer will be killed by disconnecting through the VNCserver. This is almost the same behaviour as for RECONNECT. When running in shared-mode, just an additional vncviewer will be attached to the server.

SSHPING
Not applicable.

USER
Not applicable.

VNCBASE:<base-port>
Base port as new offset for port calculations from the DISPLAY number. Standard value is 5900.

VNCPORT:<literal-port>
Port to be used literally, required for several VMs with fixed Server-Ports.

WAITC:<delay-after-viewer-call>
Delay after start of vncviewer, internally used as delay before check of PID for JOBDATA. Might not be really required to be varied, but provided for completeness.

WAITS:<delay-before-viewer-call>
Delay for start of vncviewer, required when the execution is too fast for the VNCserver to finish it's init.

The practical application could be the usage within a GROUP and/or MACRO, where for security reasons a password based access to multiple <exec-targets> is provided, e.g. for root accounts within a admin group. With setting of this parameter the initial output of VNCviewer is delayed due to it's own delay, thus a series of password requests occur without beeing poisoned by trace messages of the parallel executed VNCviewer.

BULK:[0-9]1,3
This is a bulk counter for automatic handling of given number of sessions. Mainly used for test purposes. It extends automatically the supported standard <label> with three leading-zero-digits, for each instance. Which could be DEFAULT. The default limiting maximum is set to 20.

<bulk> could be used for CREATE only.

ENUMERATE
Not applicable.

LIST
Almost the same output as common standard, with following changes in semantics.

id
The DISPLAY used by the vncviewer and/or vncserver. For the actual display of the server two cases has to be distinguished:

DISPLAYFORWARDING
The DISPLAY of vncviewer and vncserver are identical.

CONNECTIONFORWARDING
The DISPLAY of vncviewer and vncserver are different, this is due to the intermediate tunnel, which handles

the port-forwarfing and an has to do a remapping due to ambiguity within the network scope.

The following values are not applicable:

uuid, mac, tcp

Examples

Refer to Section 21.3 'VNC Examples' on page 439 .

17.3 Category PMs

This section contains help for the add-on plugins of category PMs, which are executed on native physical machines.

17.3.1 PM - Linux and BSD Hosts

Description

This plugin adds support for PM sessions to PhysicalMachines running the OS PM. As console currently Soft-Consoles are supported, which are here CLI, XTERM, and VNC.

PREREQUISITES/RECOMMENDATIONS:

1. The BIOS of the PM has to be configured appropriately. The almost in any case required configuration includes the following flags frequently found in APM mask:
   - PME Events
   - Wake On LAN - Event
   - Wake On Modem/Ring - Event

   The actual naming is similar to the above.
2. Even though this functionality should be almost HW independent, the x86_64 and i386 platform for Intel, AMD, and VIA based CPUs is tested only.
3. The required utilities are "dmiencode" and "ether-wakeup".
4. A machine record for ENUMERATE with full scope of data should be generated as root user by usage of tool "ctys-genmconf".
5. The caching database with at least a MAC mapping generated from dhcpd.conf by tool "ctys-extractMAClst" has to be present.

   Alternatively a manual entry for the destination machine could be added manually. For the format refer to "ctys-extractMAClst".

   Preferably the full database in addition should be generated by usage of "ctys-vdbgen", which is of course a collector for running systems. Therefore for bootstrap MAC mapping is required and sufficient.

Some specific requirements arise from the execution structure of CREATE for the PM plugin. This is inherent to the circumstance, that the target of PM as a container is not available before the successful execution - at least the first step - of CREATE. The specific case here is the fact, that the PM requires it's managed target as a runtime container for itself. In companion with the restricted permission to some tools like ether-wake, a gateway-host for execution of the CREATE method is required, which could be the localhost itself.

The basic instances involved into processing of PM are:

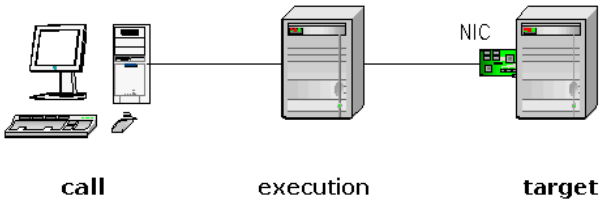

Figure 17.1: Structure of WoL configuration

The NIC on <target> requires the following pre-initialization, which is (seems) to be stored in a volatile RAM powered by stand-by power, thus has to be reinitialized when <target> is switched off.

```
"ethtool -s <ifX> wol umbg"

with Wake On:

  u - unicast messages
  m - multicast messages
  b - broadcast messages
  g - MAgicPacket(tm)
```

ATTENTION:

The interface to be addressed is the physical interface receiving the required packets. This could vary from the standard naming schema. For example in case of standard Xen the name of the physical interface name of original "eth0" is altered to "peth0".

When previous pre-initialization has been performed the following call starts the off-line, but stand-by powered <target>.

```
"ctys -t PM -a CREATE=WOL,<target> <execution>"

The machine to be activated could be addressed as:

<target> = <machine-address>

"t:192.168.1.1"
"t:myHostDNSName"
"m:00:0A:0B:0C:0D:0E"
"l:MyLabel"
```

Therefore for the CREATE(REUSE,RESUME) method of PM is currently the DISPLAYFORWARDING sub-option of "-L" supported only. This is due to the requirement of implementing a "polled-wait" on an initially booted host, which could take some time. For reasons of reduction of development efforts the relay-host - namely the <execution-target> - is used for the complete task of the action CREATE. This means in consequence as the execution base for the virtual CONSOLE to attached too. So due to usage of initially established "ssh-tunnels" only, for any kind of console, DISPLAYFORWARDING is the only applicable action. It supports any kind of terminal application applicable within an X11 environment. This limitation does not apply to the action CREATE(CONNECT,RECONNECT) which support only the attachment to already running instances, and therefore digging their own fresh tunnels. Any other action could be applied as common.

The following environment variables control the behaviour of PM plugin. For a complete list and additional documentation refer to configuration file of PM ".../pm/pm.conf.$MYOS".

1. Basic system calls, these require root permissions and therefore will be prefixed by PMCALL variable.
   The following variables are fixed, therefore cannot be altered by pre-set environment, but within the OS specific configuration file.

   CTYS_HALT
   CTYS_REBOOT
   CTYS_POWEROFF
   CTYS_INIT

   CTYS_IFCONFIG
   CTYS_ETHTOOL
   CTYS_WOL

2. Base call variants with generic options for PM specific calls. There are basically two variants of internal call, due to current design decision.
   (a) for remote WoL call, a shell based own solution is used, which is based on "Netcat". This uses UDP broadcasts.
   (b) for local WoL broadcasts the tool "ether-wake" is used, which sends Ethernet broadcasts.

   These two tools require slightly different parameters and access permissions. The following are the pre-requisites for each of the tool:
   (a) Native MagicPackets(TM) by shell scripts and forwarding by "Netcat". Support of routers, particularly the final router connecting the destination subnet. Therefore "directed broadcasts" as well as specific redirection of IP-addresses and/or ports are supported. The "-t" option could be a directed broadcast address or a specific address with an optional port.
   (b) Call of "ether-wake" for local packets only. Root-access permission for write-access to interface is required.

   CTYS_WOL_ETHTOOL
   CTYS_WOL_ETHTOOLIF

   CTYS_WOL_WAKEUP
   CTYS_WOL_WAKEUPIF

CTYS_WOL_BROADCAST

REMARK:

In current version the first interface (which is not "lo") of the CTYS_IFCONFIG call will be used by default, thus this works for bonding too. When another interface is required, than the variable should be preset/exported, e.g. by "CTYS_WOL_ETHTOOLIF=eth7".

3. Common parameters influencing the processing behaviour, these could partly be set by CLI parameters:

   CTYS_WOL_WAKEUP_WAIT
   CTYS_WOL_WAKEUP_MAXTRIAL
   CONSOLE_IPC
   CONSOLE_SERIAL
   PM_POWOFFDELAY

Options

-a <action>[=<action-suboptions>]

CANCEL

```
CANCEL=(<machine-address>)\{1\}
  (
    [FORCE|STACK]
    [SELF]
    [
      RESET|REBOOT
      |(INIT:<init-state>)
      |(PAUSE|S3)|(SUSPEND|S4)
      |(POWEROFF|S5)[:<timeoutBeforeKillVM>]
    ]
    [TIMEOUT:<timeout-value>]
    [WOL]
    [IF:<if-for-wol>]
  )
```

SELF

Self cancels the hosting machine for the call after cancel operations on the stack, if not provided the stack is canceld only. This is due to conformity with the CANCEL

operations of VMs, where the hosting system for the hypervisor is addressed. This suboption basically should be applied as default and only behaviour for a PM. But it seems to be somewhat smart, just to target anything within a PM, e.g. when bringing the PM into maintenance mode, without changing the operational state of the PM itself. Due to the main targeted user as a systems administrator, the simplicity of the UI was in this case below the advantage. So sadly accepting some irritation with this suboption here, making it "not default". When SELF is required, it has to provided.

WOL

WOL flag is supported for NIC drivers supporting the ethtool interface, which are e.g.

e1000, tg3

8139too, e100

For cards with configuration by modprobe-interface this flag is not applicable.

WOL enables the later call of "CREATE=WOL,..." by setting the volatile registers of the NIC via the call of

```
ethtool -s <ifX> wol g
```

thus WoL via MagicPacket(tm) is enabled after "halt -p" call.

For successful execution root permissions for operations on the interface by "ethertool" is required and should be configured vi sudoers or .k5users.

This state change request is not propagated to the VM-stack, the PM plugin itself is the only one evaluating it.

When the WoL feature is set persistently by means of the OS, the WOL flag is not required.

IF:<if-for-wol>

The interface to be parared for WoL. Access permission for "ethtool" is required. If this suboption is not provided, than the first detected valid interface is used.

Therefore each plugin is called by the ctys specific init-mechanism in order to prepare the shutdown, though the prepare the interfaces too.

This is particularly for bridged networks required, where the bridged interfaces not propagate the "wol g" flag properly in each case. Thus the detection of the valid interfaces is performed after the completed plugin shutdown actions.

CREATE

```
CREATE=<machine-address>
  [,(REUSE|CONNECT|RECONNECT|RESUME)]
  [,WOL[:<wol-delay>]
  [
    (
      ,BROADCAST:<broadcast-dest-address>
      [,PORT:<broadcast-dest-port>]
      [,BMAC:<broadcast-MAC-address>]
    )
    |
    (IF:<broadcast-interface>)
  ]
  [,CONSOLE:(CLI|GTERM|XTERM|EMACS|VNC)]
  [,USER:<user>]
  [,PING:(OFF|<#repetition>%<sleep>)]
  [,SSHPING:(OFF|<#repetition>%<sleep>)]
```

For information on USER , PING , and SSHPING refer to common options.

<machine-address>
: The address of the machine to be activated. The usage of <machine-address> implies an exact target to be addressed explicitly.

  For multihomed machines a minor limit occurs, when the NIC to be used for WoL is different from the NIC for usage of CONSOLE access. Currently CONSOLE suboption is applied to the target <machine-address>, which is also used as target for the WoL activation.

This can cause difficulties, e.g. when a board is used, where - due to BIOS limits - a low rate NIC has to be used for WoL activation, whereas the normal traffic is performed by the remaining NICs with channel bonding. The same principle could be applied due to security reasons when multiple NICs are available.

Anyhow, the workaround is to use two calls, one for the WOL packet without a CONSOLE, and a second one for the CONSOLE only, addressing the access port.

BMAC:<broadcast-MAC-address>
The MAC address to be inserted into the broadcast packet, which is defined to be the MAC address of the receiving NIC. In difference to the <machine-address>, which is strongly checked for consitency, the broadcast target is not checked for consitency. This is due to various remote application cases, where the actual data might noct be locally available, or a relay with port-forwarding and/or protocol-forwarding might be involved.

BROADCAST:<broadcast-address>
For machines not within the callers subnet, an arbitrary broadcast address could be set. A UDP package containing a MagicPacket(TM) is sent to the given address with port "DISCARD=9" set by default.

The Address has to be of one of the following types:

1. An arbitrary defined address with an optional port:
   <ip-address>
   This will require the final router for the target subnet to redirect the packet from the destined address to a local broadcast message on the subnet. Thus some additional configuration on the router is required.
2. A directed broadcast address, where all subnet bits are set. The user has to be aware of the netmask for the target subnet.

This parameter forces the usage of native script for generating and sending of the WoL packet.

This parameter cannot be combined with the "IF" suboption.

An advanced example is provided within the following chapter containing additional examples.

CONNECT
Opens just a new CONSOLE, requires therefore the PM in operational state.

CONSOLE:(CLI|XTERM|GTERM|EMACS|VNC)
If given a console will be opened on the site of caller. This will be performed in addition to the standard console, which may be attached to the KVM-connectors.

Default is CLI, which is based on a complete subcall with type CLI.

The following

The "-b" option will be set for the types of consoles as:

```
CLI    => "-b 0"
XTERM  => "-b 1"
GTERM  => "-b 1"
EMACS  => "-b 1"
VNC    => "-b 1"
```

IF:<broadcast-interface>
This defines the local interface where an ethernet broadcast is to be sent. Therefore the tool "etherwake" is utilized.

This is the default behaviour when neither the BROADCAST

nor the IF parameter is supported.

For the default case the first interface will be evaluated from a "ifconfig" call, which depends on the current OS and possible custom configuration.

This parameter cannot be combined with BROADCAST.

PORT:<broadcast-dest-port>
A port to be used on the remote site. A UDP package containing a MagicPacket(TM) is sent to the given BROADCAST address with port "DISCARD=9" set by default.

RECONNECT
All current CONSOLEs of the user are CANCELed, and a new one is started. The CANCEL just ignores any child process of the enumerated CONSOLEs, thus the user is responsible for proper shutdown of running subtatsks. Specific exchange of a single client is not supported.

When PM is yet running, it will be rebooted, else just booted and connected with the choosen CONSOLE.

RESUME
In this version the same as create only.

REUSE
When a nachine is already active just a connect is performed.
In case of a required new session, first the machine is booted, if an appropriate activation method is defined, which is WoL only in this version. When the machine is available, a CONSOLE is opened is requested.

WOL[:<wol-delay>]
The addressed PM will be activated by Wake-On-LAN, which is in current version the only supported method. Therefore the default "ether-wakeup" tool is utilized by default. This could be changed by setting the environment variable CONSOLE_WOL, to which the MAC-address of the target will be appended. The current implementation expects all

PMs within one subnet.

The timer <wol-delay> is the time period to be delayed the execution after sending the WoL packet. This timer is waited for once before starting the periodical poll with shorter timeout.

ENUMERATE
: This is specific for PM, just the local configuration information is displayed.

LIST
: This is specific for PM, just the local configuration information is listed.

Examples

Refer to Section 23.1 'PM Examples' on page 501 .

17.4 Category VMs

This section contains help for the add-on plugins of category VMs, which are executed on OS level in order to start and operate a specific type of VM.

17.4.1 QEMU - QEMU Emulator

Description
This plugin adds support for QEMU sessions[88, QEMU]. The current supported standard client types of this package are CLI, X11 and VNC. Dependent of the called qemu-version and the embedded OS the actual resulting support for the console could vary.

REMARK:

I will once again thank Mr. Fabrice Bellard for his impressive work. And all additional contributors of course.

Because of the almost more than complete coverage of any possible interaction interface, I had to reduce the current efforts due to priorities to a most common subset for now. In current version additionally the defined configuration interface for batch based calls of the QEMU VMs is defined as an open shell call interface with a little of required call parameters, but some specific processing and call assembly within the called script.

The required customization contains the requirement of basic scripting knowledge and at least some knowledge of the call interface including basics of the framework, but it offers therefore a maximum of flexibility and openness to the user.

A number of templates from my own test environment are supported within the standard installation and will be continuously maintained, thus might suffice for non-developer users too. This includes templates for some of the supported QEMU examples too.

Some generic templates for usage-from-the-box will follow soon.

Last but not least, the current syntax is probably going to change slightly soon, because this version is a stable and working release, but mainly a proof of concept for applications of

stacked VMs. QEMU is the vital building block for this. Ongoing experience in application of the introduced principles might lead to some variation for the application of QEMU, and therefore for the call interface.

Specifics like the attachment of gdb are not supported within the command set of ctys, even though could be configured by the config file-script and will be forwarded to caller's machine by "DISPLAY-FORWARDING".

Networking

Even though there might be other viable solutions for the networking interconnection of QEMU, the "author gave it up" to continue with trial-and-error and choose the following solution for the first version. A tap based additional design will follow for the usage of QEMU on BSD platform.

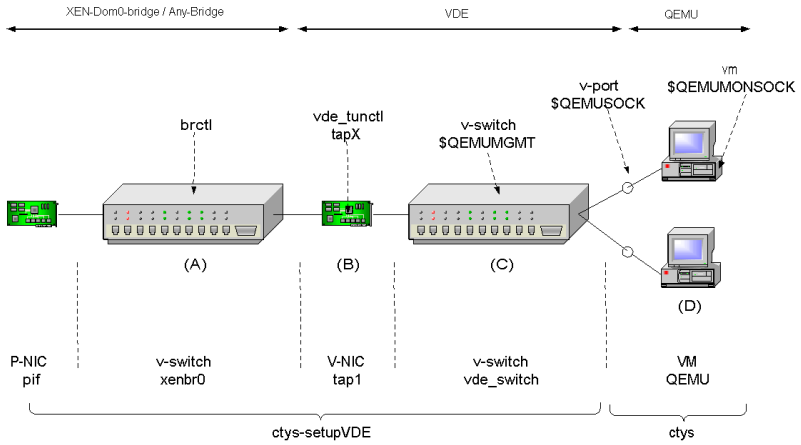

Figure 17.2: QEMU NIC interconnection

The QEMU plugin utilizes the VDE package exclusively for setting up network connections. The verified and used version is here vde2 as a mandatory prerequisite. This is due to the following two features mainly:

- Availability of tunctl, which is named "vde_tunctl" within vde. This works on the supported OSs perfectly. A TAP

device is mandatory for QEMU to be interconnected in a transparent bridging mode as applied within this version at least.

- The User-Space virtual switch "vde_switch", which just requires one tap-device to attach itself to the "external" network, whereas the users can attach to it's internal ports with their own permissions. This is permitted due to pre-assignment of it's owner by usage of "vde_tunctl".

A short description of the install process for QEMU with network support, pxe-boot/install, and cdrom-boot/support based on the examples from QEMU is given Section 22.1.1 'Installation of QEMU' on page 445 . The setup of QEMU with VDE for TAN is described in Section 22.1.1 'Setup Networking for QEMU by VDE' on page 446 . Downloads are available from [130, VDE2], a very good decscription about networking with TAP could be read at [131, VirtualSquare].

Once the basic install and setup is completed, the whole process for the creation of the required networking environment is handled for the local stack-entity as well as for remote stack-entities by one call of the tool ctys-setupVDE only. The scope of operations of ctys-setupVDE and ctys is depicted in figure:13.2.

The listed environment variables are to be used within the configuration scripts. These are particularly mandatory for to be present and accessible by usage of ctys. So the CANCEL action for example will open a connection to the QEMUMONSOCK and sends some monitor commands . The QEMUMGMT variable will be used to evaluate the related tap-device, and for final deletion of the swithc, when no more clients are present.

All sockets are foreseen to be within UNIX domain only, as designed into the overall security principle. Anyhow some minor break might occur for the vnc port for now, and should be blocked by aditional firewall rules for remote access.

The listed variables are required partly to be modified with dynamic runtime data such as the actual USER id and the PID as shown in the various examples. The definition and initialization is set in the central plugin-configuration file "qemu.conf".

- QEMUMONSOCK
  The monitoring socket within local UNIX domain, will be used as

  "-serial mon:unix$QEMUMONSOCK,server,nowait".

  Could be attached by the terminal emulations:
  - "nc -U $QEMUMONSOCK"
    "netcat -U $QEMUMONSOCK"
  - "unixterm $QEMUMONSOCK"

  It should be the first entry containing the serial console "ttyS0" too, which is for the provided ctys-examples assumed, and is the case.
- QEMUSOCK
  The socket to be attached to the vde_switch. One specific port, therefore specific QEMUSOCK is derived for each running QEMU instance.
  The network socket within local UNIX domain, will be used as

  "-net vde,sock=$QEMUSOCK".
- QEMUMGMT
  The socket for management of the virtual switch. This could be also accessed by nc/netcat and unixterm. The utility "ctys-setupVDE" widely utilizes this interface.
  - "nc -U $QEMUMGMT"
    "netcat -U $QEMUMGMT"
  - "unixterm $QEMUMGMT"

Configuration File

Due to the primarily CLI oriented and pure call-options based configuration a standard file is defined within ctys, which supports all entries for the start call and for the enumeration of available VMs. Therefore first of all a file is defined. Second the standard notation of additional configuration values is adapted to this file. So, the UnifiedSessionsManager requires mandatorily the presence of a configuration file to be able to handle QEMU VMs. The syntax of the configuration file is described in Section 33.7 ‘Configuration Entries for ctys’ on page 624 .

Therefore a standard subset of suboptions for the CREATE call is provided. This is similar to the other supported VMs. The syntax of this file concerning the value assignments is

described in the Appendix. In addition it is an executable script, which performs some final evaluation of parameters to be passed to QEMU. This is pure bash-shell syntax. The naming convention is as for all. For most of the QEMU provided test VMs a suitable configuration file is supported under the ".ctys/templates/qemu" directory.

The expected directory structure for assembly of the runtime call is as given:

```
$HOME
+-.ctys
| |
| +qemu       QEMU specific configuration files for user specific
|             settings, else the installed are used as default.
+-qemu
| |
| +-ctys      New VMs.
| |
| +-pc-bios   Supported bios files, e.g. pxe/ether-boot.
| |
| +-tst-ctys  Supported ready-to-use test VMs of QEMU with
| |           ctys-config-files
| |           These files could be used as templates for
| |           new VMs.
| |
| +-qemu-example-wrapper
|             Supplied demos accompanying the QEMU distribution.
+-ctys
  |
  +-templates.dir
  |           Template directory tree to be used for $HOME/....
  |
  +-templates.vm
              Template configuration files to used or referenced
              as examples.
```

Figure 17.3: UnifiedSessionsManager QEMU file structure

Supported CPUs

The whole set of QEMU's CPUs is supported[89], which includes for version 0.9.1:

x86, AMD64, ARM, MIPS, PPC, PPC64, SH4, M68K, ALPHA, SPARK

Same for a number of supported OSs[91].

The appropriate call has to be configured within the appropriate configuration file. Templates as ready-to-use templates for the provided QEMU tests are included for x86, Arm, Coldfire, and SPARC. Running Linux, uCLinux, and NetBSD.

Supported Interfaces

Qemu supports various interfaces for interconnection of it's hosted GuestOS to an external device. Particularly the HMI applicable interfaces for CONSOLE interconnection are of interest for the QEMU plugin as a hypervisor controller, whereas the support for native is handled by the HOSTs plugings, refer to Section 17.2 'Category HOSTs' on page 276 .

The encapsulation of the interface for access from "outside" to the "inside-GusteOS" is encaspsulated by the QEMU-VM via usage of specific virtualisation drivers. These drivers actually manipulate the payload-dataflow, whereas the outer encaspsulation by the UnifiedSessionsManager is a control only encapsulation.

The outer encapsulation by the UnifiedSessionsManager is divided into two parts, the custom wrapper-script for finally executing the QEMU VM by calling the VDE-wrapper, and the call options for specific CONSOLE types, which prepare some additional required execution environment for the various call contexts. It should be obvious that the two outer encapsulation components has to cooperate seamless.

For the actual and final interconnection to the GuestOS the CONSOLE types SDL and VNC are based on a transient virtual HW, thus almost need no additional configuration, but the activation for the QEMU VM. SOmewaht different is the behaviour of usage of a text-only console, which is in case of UNIX attached to a serial device, a so called ttyS. This requires some additional native configuration of the GuestOS, in order to attache the chracter stream within the domain of the GuestOS to the outer interconnector. For a detailed example refer to Section 22.1.1 'Setup Serial Console' on page 450 .

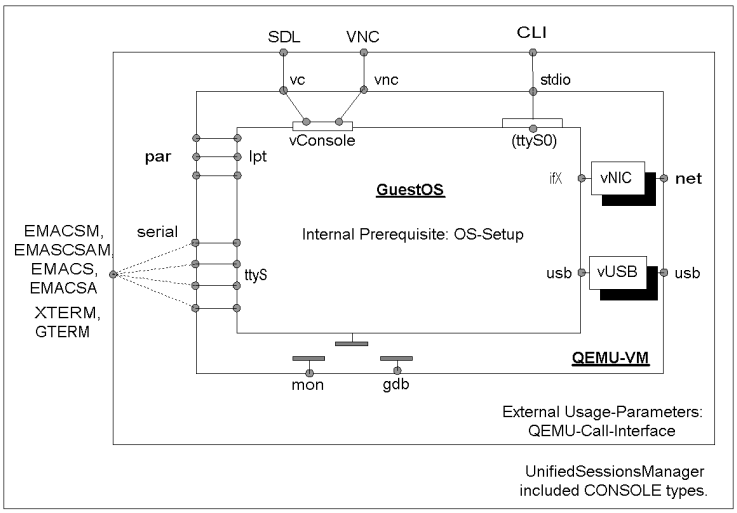

Figure 17.4: QEMU Interconnection-Interfaces

The following description is not yet finally verified, but works within the UnifiedSessionsManager. Some statements might need to be corrected, when experience and understanding of the whole scope of interfaces for QEMU is somewhat extended.

- monitor
  The QEMU monitor port is supported as a local UNIX-Domain socket only. The socket name is assembled by a predefined environment variable and the PID of the master process for the final wrapper script , which is executing the QEMU VM and has to be configured by the user.

  For the various CONSOLE type different handling of the monitor port is applied:

  CLI0
  : No specific port, stdio is used for console as well as for monitor.

  SDL, VNC
  : Extra monitor port QEMUMONSOCK, used as multiplexed port, could be connected additionally by an ASC-II console.

  CLI, XTERM, GTERM, EMACS, EMACSM, EMACSA, EMACSAM
  : Mapped monitor port in multiplex mode on UNIX-Domain socket QEMUMONSOCK for re-attacheable console port.

The base variable is "QEMUMONSOCK" , which contains by convention the substrings "ACTUALLABEL" and "ACTUALPID". These two substrings will be replaced by their actual values evaluated when valid during runtime. The "ACTUALLABEL" is the label of the current VM, as will be provided to the commandline option "-name" of QEMU. The "ACTUALPID" is the master pid of the wrapper script, which will be evaluated by the internal utility "ctys-getMasterPid" . The master pid is displayed as the SPORT value, even though it is used as a part of the actual UNIX domain socket only.

The default socket-path is:

"/var/tmp/qemumon.ACTUALLABEL.ACTUALPID.$USER"

This will be replaced e.g. to:

"/var/tmp/qemumon.arm-test.4711.tstuser1"

Any terminal application like "unixterm"[131, VDE/VirtualSquare] of VDE package, or "netcat/nc"[141, GNUNetcat][142, NetcatWiki] could be used for interaction. The switch between QEMU monitor and a text console is the same as for the "-nographic" mode by "Ctrl-a-c", for additional information refer to the QEMU user-manual[90, QEMUDOC].

The monitor socket is utilized by internal management calls like CANCEL action by usage of "netcat/nc". The following controls are used for monitor:

Ctrl-a	001
x	170
c	143
S3	stop|cont
S4	savevm/loadvm[tagid]
S5	Ctrl-ax

Table 17.2: Utilized QEMU-Monitor-Commands

- net
  Network devices, here used as classical network devices only in combination with VDE as wrapper.

  Could be used for various access to native OS, some specific QEMU VM terminal emulations are supported, refer to the "QEMU User-Manual" [90].

- par[0-2]
  ffs.

- sdl
  The SDL is attached to a virtual console driver, which emulates the Key-Video-Mouse of a real machine. It is present in any case and active by default, but could be deactivated by usage of "-nographic" parameter. This is the workhorse, because it is available by default and does not require any GuestOS pre-configuration. An alternative may be the VNC.

  The following statement reflects the authors current state of understanding only and could be still faulty:

  The SDL console seems to work synchronous only, and terminates the QEMU VM as ssoon as it's window is closed. Therefore this mode could not be used in the initial headless mode nor for a later detachment and reconnect.

  Multiple CONSOLES for prevention of terminating the QEMU VM are not yet evaluated. Also not yet evaluated, whether a socket could be probably utilized for some "persistence".

- serial[0-3]

  QEMU supports by default up to 4 of serial devices. Due to priority on operability for the UnifiedSessionsManager, one port is forseen for CLI, VNC, and SDL mode, two serial ports are foreseen for the remaining modes to be used by the framework.

  For the CLI console no extra monitoring port is alloacted, the default values for "-nongraphic", which are stdout/stdin

with a multiplexed monitoring port, are used.

One serial device is reserved for an additional monitor port exclusively for the types SDL and VNC.

For the remaining CONSOLE types, which are variants of CLI type, the monitoring port is multiplexed to the console port again, but now for an allocated common UNIX-Domain socket.

This port is required in order to open a management interface. When suppressed some actions like CANCEL may not work properly. This is for example the case, for the final close of the stopped QEMU VM, which requires frequently a monitor action.

-serial mon:unix:$MYQEMUMONSOCK,server,nowait

- stdio
  This attaches the console to the callers shell. Requires preconfiguration of a serial device within the GuestOS, for a template refer to Section 22.1.1 'Setup Serial Console' on page 450 .
- usb
  ffs.
- vnc
  This console replaces the SDL type when choosen. It works as a virtual Keyboard-Video-Mouse console by default and thus does not require pre-configuration of the GuestOS. But needs to be explicitly activated by the "-vnc" option.

Some additional remarks/reminder

- When terminating a CLI session, the prompt will be released by a monitor short-cut: "Ctrl-a x".

  In some cases a "Ctrl-c" is sufficient.

Options

-a <action>[=<action-suboptions>]

CANCEL
Specific semantics for SPORT within QEMUMONSOCK as described before.

The CONSOLE type CLI0 requires specific handling for CANCEL.

CREATE
All standard parameters not listed here could be applied.

BOOTMODE
For BOOTMODE multiple configuration variants are supported. In any case only one of them is supported to be the active "master" at any time. This is controlled by "MAGICID-QEMU" and "MAGICID-IGNORE" , which incfluence the configuration-detection process of ENUMERATE, and therefore the caching behaviour too. In case of QEMU this is the file containing the configuration keys to be used.

Conf-File	Behaviour
ctys	Contains ctys specific configuration keys by prefix "#@#" and assembly of QEMU call, which is finally executed
ctys+conf	The same as ctys, but the configuration keys are "sourced" from an external conf-file, which is required to be coallocated within the same directory. Templates are available.

Table 17.3: Supported Configuration files for QEMU

REMARK:
The executable "*.ctys" requires the x-bit to be set.

CONSOLE:CLI
The CLI mode is the backed by the plugin CLI, which is utilized in the same manner as the other X11-based plugins, thus attached by usage of the UNIX-Domain socket QEMUMONSOCK for a serial port. The CLI console is detachable and could be re-attached later.

CONSOLE:CLI0
This mode deviates from the common CLI mode, and is tightly coupled to the VM, thus could not be detached. When the console is detached, the VM will be terminated.

Therefore this mode sets particularly the following options and operational modifications.

- trap: INT,TSTP, QUIT
  These signals are deactivated in the first instance of the called client, and in the entry instance of the remote client(s). Thus the signals, if activated, are transparently passed though to the target peer. The values could be configured by the variable and/or set by the option "-S" .
  "CTYS_SIGIGNORESPEC".
- "-b 0,2 -z 2"
  This mode sets implicitly "-b 0,2 -z 2", otherwise the the input stream might be disconnected. The background mode is generally not applicable to CLI0.

When the GuestOS is shutdown in CLI0 mode the console stays still occupied by the QEMU VM after the guest system is halted. In order to release the CONSOLE/Terminal, the monitor has to be used. Call "Ctrl-A-c-<RETURN>", and when the "(qemu)" monitor prompt occurs, execute "quit" within the monitor.

CONSOLE:SDL
: This is the standard graphical console of QEMU.

PXE
: This boots an image into PXE, currently only for x86 platforms supported.

LIST

Due to various output some assumption has to be made for processing. One point is, that for now the output of "ps" command is analysed for retrival of the active sessions.

The other aspect is the access permission to files which are not owned by the caller. So it is a design decision to bring as much information as possible to the command line.

Is this a security flaw? Maybe, but the main aspect of ctys tools is to prevent the local access to a ps call, if the caller is not welcome. Therefore the information within the ps-output is not seen as more compromising, as the UNIX system is anyway or not.

One important definition is the usage of vde tools as a general wrapper for qemu. This is due to the main target of production and software development usage of Qemu, more than HW development. Therefore a easy to use userland wrapper for "TAN" device is required. The userland switch and the vde_tunctl are what is required.

The consequence is that the "vdeq|vdeqemu" strings are used as key for scan of ps-output. Thus native qemu calls will not be matched.

Due to transformation information into ps-output the MAC address will be used in any case, which might not be much more than required anyway.

For the same reason the label, which is a valid selector, has to be identical with the directory name, in which the QEMU-VM is contained.

Another convention to be fulfilled is that the last pathname of the call has to be related to the VM, "-L" option and other pathnames have be left-of. This is tremendously simplifying any regexpr match and should be for the developer of a configuration file easy to handle.

One more repetition, the following items have specific requiremenents and semantics, which has to be recognized by a custom wrapper script.

PID
: This is the PID of the VDE/VirtualSquare wrapper "vdeq", and will be handeled and required by the string processing routines scanning the "ps" output.

SPORT
: This is the master-pid, which is the pid of the top-level custom wrapper script.

REMINDER: Spaces in any entry are not supported.

The following split between dynamic and static retrival is made:

ps-only
: PID, UID, GID, DISPLAY, MAC, LABEL, SPORT

config-file
: UUID

miscellaneous
: ID, HOST, SESSIONTYPE, CPORT, CSTYPE

-g <geometry>|<geometryExtended>
: The geometry could be set for the clients only, the resolution parameter "-r" is not applicable:

  CLI
  : Not applicable.

  SDL
  : Limited applicable, not yet supported/tested, will follow soon.

  XTERM|GTERM
  : The size Xsiz and Ysiz provide the UNIT of CHARACTERS only.

  VNC
  : As expected.

-r <resolution>
: Not supported.

Examples

Refer to Section 22.1 'QEMU Examples' on page 445 . Particularly the installed templates and example configurations might be helpful. As a first entrypoint us the configuration for the QEMU supplied examples.

17.4.2 VMW - VMware

Description
This plugin adds support for VMW sessions. The current supported client type of this package is the native GUI provided by VMware.

Any non-listed standard option of ctys applies as defined. In cases of divergent behaviour for similar options, and options with specific suboptions are are listed in this section.

REMARK:

Even though the products of VMware are not OpenSource, at least not as a whole, I personally own several commercial and free licenses, use them frequently and don't want to miss them.

The products are used perfectly in almost the same manner as a so called "not-open-too-PC" on a unified Linux-Based distributed network. Where the network IS actually the COMPUTER and almost anything else than Linux is just a "roamed virtual device" within a "dynamic VLAN".

This is completed by OpenBSD as the choice of the diskless "embedded" networking equipment.

Therefore I decided to forward this bridging plugin as one of the building blocks for some other upcoming projects.

The most important applications of VMware are the

final migration of Windows-Applications as hardware and location independent appliances,

and

setting up a Cross-Build environment on machines without hardware based virtualisation support.

Current supported products are:

VMware-WS	(latest ctys versions tested with WS6+)
VMware-Server	(tested with 1.0.[34], 1.0.5+ might work)
VMware-Player	(tested with 1.0.5, 1.0.6+ might work)

PREREQUISITES/RECOMMENDATIONS:

1. background operations The background operations for the server component of the VM has to be set explicitly for some variants. This is not required for server-products, but should be done for workstation products.

   Not setting this leads to an immediate termination of the server, when client closes. Not necessarily with a soft-shutdown!!!

2. tabs-mode
   The tabs-mode for the proprietary CONSOLE should be set off. Even though a tabbed-view could be used too.

   Due to the embedded dispatcher for the display the CONSOLE requires here a user interaction for selecting the target display in any case(by tabs, or by menu "tabs"), but when only one display per window is assigned it appears to be little more straight-forward.

3. authentication
   User authentication is for "-P <port>" access even for "-h localhost" required, so for CONNECTIONFORWARDING in any case the user seems to have to perform a login. In addition, it seems that the user has to be a local user on that machine.

   As far as I can say for now, only in case of DISPLAYFORWARDING SSO has an effect on sessions. Let me please know, when this is wrong, and don't forget to explain how.

4. Headless-Start
   Found a "nogui" param for vmrun, will be introduced asap.

Options

-a <action>[=<action-suboptions>]

CANCEL
Following apply:

CLIENT
The client server communications is handeled by a proprietary frontend, which implements the sharing of a single port, by default 904. This requires some specific

treatment particularly for the case of CONNECTION-FORWARDING.

The headless-mode(NONE) for CONSOLE is not supported as initial call, the console could be detached without shutting down the server component later, when background-mode is preconfigured.

This is not applicable for VMplayer.

SERVER
This is not applicable for VMplayer.

BOTH
When this keyword is present the server and client processes of the previous set are filtered.

CREATE
VM or Team to be loaded and options passed through. For additional information refer to the related documentation from VMware. E.g. "Workstation User's Manual " Workstation 6.0 Appendix A "Workstation Command-Line Reference"; pg. 403.

<callopts>
When <callopts> are given, these will be passed through to the call:

vmware <callopts> <vmx-path>

The "–" double hyphen is inserted as required.

Be aware, that some of these such as "-geometry" and "-name" are already implicitly utilized by other options, thus use this if, than CAREFULLY.

CONSOLE

All available console types for the supported VMware versions could be used, but the specific support of the versions may vary. An overview of provided CONSOLE types is listed within the description of the standard call for CREATE.

Particularly the following console types are available for VMW:

- VMW
  The standard frontend, is available for all. When configured within the VM a detach and re-attach to the running VM is possible. The background mode is required here, which is not available for at least some versions of the Player.
- NONE
  The initial headless mode, available for Server and Workstation, requires some specific configuration of the VM.
- VNC
  The use of VNCviewer as frontend, available for Workstation-Version-6, requires configuraition of VM. The port has to be configured statically within the VM, but is evaluated dynamically by mapping of the <machine-address> at runtime.

ENUMERATE
Enumerates all VMW sessions, therefore the vmx-files will be scanned and the matched attributes displayed.

Therefore the following order of files will be scanned for values:

1. <pname>
   The standard configuration file for VM, as given.
2. <pname-prefix>.ctys
   The prefix of given filename with the ".ctys" suffix.
3. <pname-dirname>.ctys
   The dirname of given file with ".ctys" suffix.

In each case the searched key is expected to have the prefix "#@#" within the file.

-g <geometry>|<geometryExtended>
The geometry has a slightly different behaviour to the standard when specific options of proprietary WS-CONSOLE are switched on.

- The positioning parts of parameters of <geometryExtended> seem to work in any case correctly.
- The offset of <geometry> seems to work proper in any case too.

- The size of <geometry> seems to work proper when "AutofitWindow" and "AutofitGuest" are switched off.

  As could be seen, shortly after start of CONSOLE it will resize itself, if previous parameters are set.

  Which is indeed a pretty well behaviour. What else should that options of CONSOLE control?

-L <execution-location>

```
<execution-location>=(
   (LOCALONLY|LO)
   | (CONNECTIONFORWARDING|CF)
   | (DISPLAYFORWARDING|DF)
   | (SERVERONLY|SO)
  )
```

Currently the following selections are supported:

	LO	CF	DF	SO
VMware-WS		no	yes	1)
VMware-Server		yes	yes	1)
VMware-Player				

Table 17.4: Forwarding modes and call locations for VMW versions

1) The background-server-mode is currently implicitly supported only.
   This requires the options to be selected within the products - which are slightly different, but are almost commonly supported - and will be implicitly started only when starting the whole product.
   The server component continues execution when the client is canceled, but could not be started separately.
   A CONNECT to a running server is supported.

Examples
Refer to Section 22.2 'VMW Examples' on page 464 .

17.4.3 XEN - Xen

Description
This plugin adds support for XEN sessions. The current supported client type of this package is CLI , X11 , and VNC.

Any non-listed standard option of ctys applies as defined. In cases of divergent behaviour for similar options, and options with specific suboptions are are listed in this section.

There are some specifics to be recognized and/or applied specific to Xen. This is primarily due to it's nature of the hypervisor interface, where DomU-s are children of the one and only Dom0, which is not "visible" to "ps" as a normal process, but to the specific tools "xm" and "virsh". Where virsh is part of "libvirt" but prerequired for ctys.

One main challange in combination of access to restricted system resources is the requirement of root permissions for some calls to manage DomU-s. This requires for user-level on demand CREATE and CANCEL the configuration of "sudo" or "ksu".

Some drawbacks for the common applied tricks of ctys, using the CLI and "ps" as an dynamic storage and exchange interface for runtime information are not working in the altered runtime environment. Even though particularly the "virsh dumpxml" call offers a variety of information. One missing data, that really "hurts" is the missing information of the used configuration file for the list-ed or dumpxml-ed domain. The "source file" is available - which is the virtual boot-device. But this does not allow an back annotation to related configuration file - this could be just safely defined by additional naming convention, what is done within ctys for simplicity.

Another specifics is a legacy of ctys, which is the definition of ID as a static unique identifier for a VM and PM entity, which does not change, when the entity changes it's state to offline. Resulting of this, the ID is for VMW , PM, and XEN the fully qualified pathname of the configuration file, which is still not necessarily unique, due to NFS mounted shares on multiple PMs and/or VMs. This is still not unique, when combining the PMs hostname and the pathname of the configuration file, because the contained IDs, e.g. TCP/IP address, MAC address, and UUID are now available within multiple entities, and thus will be listed as though when using administrative management utilities. Anyhow, it should be at

least guaranteed by the user, that the entities are unique within the scope a single node. The toolset is prepared to handle various constellations, but it makes the selection by the user somewhat easier.

For this the following shortcuts and conventions apply.

- The Domain-ID as provided by Xen is for now ignored, the Domain-Name is required to be unique, so the LABEL, which is the Domain-Name, is sufficient as selection criteria. This is anyhow a static constant identifier, which is not true for the Xen-Domain-ID.

  The Domain-ID within ctys - "IDS" for ctys-vhost - is a holomorphic identifier, which is for machines - VMs and PMs - a configuration filepathname, for types of the category HOSTs a dynamic system generated ID such as a PID, DISPLAY, or port.

  Therefore the Domain-ID for Xen within ctys is the filepathname of the configuration file. This is particularly important due to stored information within the configuration file itself, or within the same directory. Due to the only available filepathname for the boot-image of the DomU instance by "virsh", the fixed - maybe already widely applied - convention is defined, that the configuration file has to be coallocated within the same directory as the virtual boot device for the DomU and to be named the same as the name of the containing directory. This has not necessarily to be the LABEL which is the Domain-Name of the DomU, but could be. SO boot devices, which are physical, not virtual files, are not supported for now.

- NO SPACES within ANY entry are supported.

- When multiple LAN interfaces are configured, the MAC-addresses are indexed by their actual order-increment, beginning from 0. These are permuttated with any provided TCP address of the same index. E.g. MAC0 => eth0 => IP0=10.1.1.1, IP0=11.1.1.1.

- Due to the variety of consoles - CLIENTS - which could be attached and are not simply correlated, the LIST action only displays the SERVER components, which are Dom0/DomU, the clients has to be listed by an extra call to CLI, X11, and/or VNC.

- The execution of the creation by "xm" and some "virsh" access permissions has to be activated and required to be with root

permissions. Therefore the configuration file "/etc/sudoers" and/or "/root/.k5users" has to be configured. The access privileges by "sudo" and "ksu -e" will be checked and set appropriately. The variable "XENCALL" and "VIRSHCALL" could be preconfigured.

- The execution of XEN requires in any case the VNC module.
- The version supported by XEN is the 3.x version. The tested and verified version is Xen-3.0.3 of the CentOS-5.0 distribution, even though any 3.x version might work. The version evaluation is done by usage of rpm or xm or virsh or xmtrace. The installation paths are evaluated by which call and should be prepared for execution by PATH.
- Due to the warning-output of some tools, this is fetched as ctys WARNING, which could be fully activated by "-d" option. Particularly the "D_SYS" debug-level, which traces all system calls, might be helpful for tracing permission settings.
- The XEN plugin is stack-aware, though prepared to propagate CREATE and CANCEL actions, same for LIST.
- XEN_CONSOLE_DOMU_INIT_WAIT This variable contains the sleep value after "xm create ..." and before calling a "gnome-terminal" or "xterm". Therefore in case of a machine which has difficulties due it's performance the value could be adjusted. The current value of 8seconds seems to be safe for initialization of created DomU.

Templates and recipes for installation of Xen on CentOS-5 are available in Section 22.3.2 'Installed Systems' on page 483 .

Options

-a <action>[=<action-suboptions>]

CANCEL
full <vm-address> range is supported.

CREATE
All standard parameters not listed here could be applied.

Dependent on the choosen parameter set some specific CONSOLE types can - whereas some cannot - be applied.

CONSOLE:(CLI|XTERM|GTERM|EMACS|VNC|NONE)

The appropriate settings for the choosen console has to be prepared within the related config file.

The default CONSOLE could be pre-set by the variable XEN_CONSOLE_DEFAULT in the xen.conf file. The original default is "XTERM". The recommended text console for Gnome is GTERM, but any other could be set as default too.

	CLI	XTERM GTERM EMACS EMACSM EMACSA EMACSAM	VNC	NONE
CONNECTIONFORWARDING	-	-	X	X
DISPLAYFORWARDING	X	X	X	X
SERVERONLY	-	X	-	X
CONNECT	X	X	X	X
RECONNECT	X	X	X	X
REUSE	X	X	X	X
RESUME	X	X	X	X
"-b 0" - foreground	B+M[1]	A	A	X
"-b 1" - background	-	X	X	X
"-z 2" - pseudotty [2]	M[3]	X	X	X

Table 17.5: Applicable forwarding modes and call locations for XEN

```
X)   Supported.

A.)  Supported, but will block the call-terminal
     for the whole session, so might not be used
     from a single-console environment.
B.)  Blocks the console for other calls, thus
     allows for bulk targets serial execution
     only.
M.)  Mandatory, but could be suppressed with
     "-z NOPTY" when a terminal with some
     drawbacks is sufficient. One of specific
```

[2] Mandatory for CLI interactive shells, is forced by default. For native CLI only when no CMD provided.

[3] Mandatory for for usage of sudo if not configured otherwise.

[4] Mandatory for CLI interactive shells, is forced by default.

```
than is that the password will be echoed
for some systems in cleartext, anyhow as
a lonely night-rider it might not hurt
you.
```

Types of CONSOLE to be applied depends on the "-b" parameter for background execution too.

The following behaviour applies:

"-b 0" - synchronous foreground execution
: In this mode the current execution thread is performed synchronous in the foreground, this means particularly a CLI based console cannot be detached, when multiple tasks are in the queue in order to begin the next. Thus it would result to blocking the remaining sessions until the current has been finished by the caller.

  This parameter is allowed to be applied, but the caller has to be aware of the drawbacks, when choosing multiple execution targets.

"-b 1" - synchronous background execution
: In this mode the DomU will be started by different means for XTERM and VNC only.

  CONSOLE:CLI
  : Will be generally rejected, because multiple execution targets cannot be handled by a single physical console, and one target could be perfectly handled by "-b 0".

  CONSOLE:GTERM
  : The "gnome-terminal" which is currently simply mapped to XTERM.

  CONSOLE:XTERM
  : Starts first an xterminal by using the X11 module and initiates the startup of the DomU within the Xterminal session as a native and synchronous call to "xm -c ...". So it is basically the asynchronous variant of a CLI call.

  CONSOLE:EMACS
  : The "Emacs" is started in shell-mode, this supports

the full scope of edit features on the output buffer. The basic principle is similar to any X11 console with an embedded CLI interface.

CONSOLE:VNC

This case is somewhat different to the previous, in the way that two independent calls for the DomU itself are required.

1. The DomU has to be started, which is performed by calling "xm <conf>".
2. The VNCviewer has to be attached to the offered port by the DomU.

   Therefore a timeout will be applied, which could be controlled by the environment variable "XEN_CONSOLE_DOMU_INIT_WAIT", which is used for a sleep-call.

   So, due to buffer handling some console messages could probably be lost.

   The client call is in internal call to VNC plugin, which is basically completely independent and could be applied separately.

CONSOLE:NONE

No console is started, any type could be connected later.

"-z 2" - force allocation of a pty by ssh

Allocates a pty.

<callopts>

When <callopts> are given, these will be passed through to the call:

"xm [-c] <conf-path> <callopts> "

For additional information refer to Xen manual.

-g <geometry>|<geometryExtended>

The geometry has no effect on the server started within the DomU. Just the client will be set:

CLI

Not applicable.

XTERM|GTERM

The size Xsiz and Ysiz provide the UNIT of CHARACTERS only.

VNC
As expected.

-L <execution-location>

```
<execution-location>=(
   (LOCALONLY|LO)
   | (CONNECTIONFORWARDING|CF)
   | (DISPLAYFORWARDING|DF)
   | (SERVERONLY|SO)
  )
```

-r <resolution>
Not supported.

<xopts>
Refer to common options parts description.

Examples

Refer to Section 22.3 'Xen Examples' on page 479 .

17.5 Category GENERIC

17.5.1 LIST Plugin-Collector

17.5.2 ENUMERATE Plugin-Collector

17.5.3 SHOW Plugin-Collector

17.5.4 INFO Plugin-Collector

17.6 Category Internal

17.6.1 DIGGER - Forwarding Encrypted Connections

Encrypted Tunnel Support

The CORE plugin DIGGER is responsible for handling of connections and the assembly of appropriate prefixes for each SSH-execution. Therefore it could be seen as the manager for remote connections and their encryption. The DIGGER also contains the functions for the management of encrypted port-forwarding.

In case of local calls with the same user-id as the caller only some PATH extensions and a "cd $HOME" are prefixed, while for remote calls two additional cases has to be handled.

DISPLAYFORWARDING
: In case of DISPLAYFORWARDING an ordinary "ssh" prefix with appropriate options is generated. No specific communication and link resources are required, anything is handled by SSH software component after the execution call.

  The call is generally mapped to "$USER@localhost" when no different user is requested. Thus a call with the "$USER@localhost" or without <execution-target> arguments is handled by the internal call processor identical.

CONNECTIONFORWARDING
: In case of CONNECTIONFORWARDING some additional link establishment effort and communications resources are required.

  In this case the final communications resources include a client process on the caller's machine, a server process on the server machine, which could be of type VM or PM. In addition a link resource, the port forwarding tunnel is established. This tunnel is no longer managed by ctys once established, but is created as a ONE-SHOT connection, which terminated and releases

it's resources once a communications peer terminates.

Even though it is an implicit internal CORE functionality, the resources should be visible within LIST action with their located local and remote ports, including the user id of the owner. Therefore the DIGGER plugin supports a LIST component, which integrates as a minor sized system plugin into the "LIST Plugin-Collector".

LIST Tunnels

One additional feature of DIGGER plugin is the support for LIST action of current active tunnels. These are listed within the standard LIST output and marked as "SSH(<creator-plugin>)" with the client-server flag "T".

The ID of a tunnel is assembled from the local port and the remote port as "<local-port>-<remote-port>". Thus supporting some "speaking" information.

The tunnel is visible on the client machine only.

Chapter 18

Support Tools

18.1 ctys-dnsutil

Usage:

```
ctys-dnsutil [options] [<dns-server-list>]
```

Description:
ctys-dnsutil supports the display of data which is mainly based on the data requested from DNS by usage of "host -l <server>" call. The only current application of this utility is to generate lists as input for additional processing or display. Extended queries are supported by "ctys-vhost" utility. It has to be recognised, that not all machines might be handled by a reachable DNS server. This is particularly true for VMs located within host-only-networks performing on isolated networks by means of routing.

One important application is the usage of this tool for the LIST action of plugins from PMs. The output list will be used as initial data set for actual available active PMs.

Additional constraints related to actual runtime-state for members of raw-list will be applied for various tools and several post-analysis.

The most basic checks are based on ping and ssh access checks, but the type of the machine - PM or VM - and the hierarchy could be evaluated by several approaches.

For an initial definition and assignment of a managed PM the "ctys-genpmconf" utility has to be executed. The generated data from the directory /etc/ctys.d/pm.conf is used as an final proof, that the polled TCP/IP-address is related to a PM.

A VM is defined as a contained VM characterised by it's configuration file, which is in the case of current supported VMs an ASC-II file with specific syntax.

Options:

-c
Uses "ctys-vhost" for PM/VM evaluation instead of polling the real instance. The basic implicit access checks for ping and ssh are still performed.

The databases for ctys-vhost has to be prepared, thus not usable for initial scan to generate that databases of course.

-C
Basically the same as "-c", but here no implicit dynamic checks are performed at all. The only dynamic evaluated data is the query of the DNS server.

-d <debug-level>
Same as ctys.

-h
Print help.

-i
Show numerical TCP/IP-Address. This is supported in combination with "-n" option only without "-X" option.

-l <USER>
Remote user to be used for network logins.
DEFAULT=CTYS_NETACCOUNT(DEFAULT->USER)

-n
Show TCP/IP-Address as name. This is supported in combination with "-i" option only without "-X" option.

-R <runtime states>
Restricts a set of multiple results with additional constrains for output.

Only the possible targets which are operable or actually operational are listed. This includes the actual running VM with it's hosting PM, and in addition all

other operational machines, where the current VM is available too. This case is the most common for NFS based exec-pools, where a single VM could be accessed remotely by a number of PMs. This particularly offers the advantage of copyless-migration of online and offline VMs.

Very handy, and in companion with others probably one of the most important internal top-level-calls for GuestOS-Command-Execution.

<runtime states>=[(REVERSE|R|-),]PING|SSH[,PM|VM]

REVERSE|R|-
This reverses the resulting set, thus the "not matching" states only will be shown.

PING[:<packetcnt>[%<timeout>]]
A RUNNING session is simply "ping-ed".

Resulting statement of operational mode may result on applied security policies, but not exclusively on the state of the running OS's IP-stack.

SSH
A RUNNING session is additionally checked for SSH-connect by a dummy-interconnect.

On some nodes the timeout my take some time, so be patient when such a node is in the DNS query.

This option might be really senseful, once SSO is established and probably a common net-access-user with limited permissions for probing-only is configured.

"ssh" is the only and one state, which is an almost viable confirmation for the ability of establishing ctys connections.

PM
Checks whether machine is a PM. Therefore first

SSH-check is activated and performed, and on the remaining set of accessible machines the PM-check is performed.

PM accessibility is defined as the accessibility of the running OS on PM and the presence of the file "/etc/ctys.d/pm.conf".

VM
Checks whether machine is a VM. Therefore first SSH-check is activated and performed, and on the remaining set of accessible machines the VM-check is performed.

VM accessibility is defined as the accessibility of the running OS on VM and the presence of the file "/etc/ctys.d/vm.conf".

-V
See ctys, version output.

-X
See ctys, terse for machine output.

Arguments:

<dns-server-list>
Any DNS server to be used in "host" call.

Exit Values:

- 0: OK
  Result is valid.
- 1: NOK:
  Erroneous parameters.
- 2: NOK:
  Missing an environment element like files or databases.

18.2 ctys-extractAPRlst

Usage:

```
ctys-extractAPRlst [options] \
   <group/machine-address-list-in-same-segment>
```

Description:
ctys-extractAPRlst generates a sorted list of 3-column table containing:

<nodename>
<IP-Address>
<MAC-Address>

Alternatively output to stdout could be generated for usage in "/etc/ethers".

<MAC-Address> <nodename>

or

<MAC-Address> <IP-Address>

Therefore this tool uses a list of targets which has to be in the same segment as the executing machine. The list could contain any valid ctys <machine-address> or any <ctys-group> definition.

The entities will be resolved and accessed as selected by options. The two methods are "ping" which assures the basic TCP/IP-stack access, and "arp" which lists the mapping as required.

So the usage of this tool could be said providing "real-in-time-data", without the limitation to static configured MAC-IP mapping, so recognizing pool-addresses too. But

it's application is limited by the area of a single segment.

For additional segments the "ctys-extractMAClst" tool should be used, which relies on configuration data for DHCP servers.

ctys-tools generally expect for input generated files with default column-order, which is defined by "-n".

Options:

-E
: Generates output for "/etc/ethers". This is required when using ether-wake for WoL, what is no longer supported by ctys.

  The options "-n" and "-i" become in combination with "-E" a slightly different semantic:

  -n output is "<MAC-addr> <DNS name>"
  -i output is "<MAC-addr> <dotted IP address>"

  The option is not supported with "-E".

-n|-i|-m

  -n Print sorted records: <name>;<MAC>;<IP>
  -i Print sorted records: <IP>;<name>;<MAC>
  -m Print sorted records: <MAC>;<name>;<IP>

-h
: Print help.

-p <db-dir-path>
: Directory for data to be stored. This is - and has to be in this version - the same directory as used by ctys-vdbgen and ctys-vhost.

  So each file-based ctys-DB requires it's own mapping file for now. This is going to be extended when the LDAP based addressing is introduced. Anyhow, this feature will remain, because it has some advantages for daily business when setting up small to medium networks or multiple test-environments.

  The hard-coded filename is "macmap.cfdb"

-P
Almost same as "-p", but takes default ctys-file-DB, provided by DEFAULT_DBPATHLST.

Default output is stdout.

-V
Version.

-X
Terse output format, effects "-V" when set left-of.

Arguments:

<ping-hostlist-in-same-segment>
Any host within the same segment, will be ping-ed and arp-ed.

Exit Values:

- 0: OK
  Result is valid.
- 1: NOK:
  Erroneous parameters.
- 2: NOK:
  Missing an environment element like files or databases.

18.3 ctys-extractMAClst

Usage:

```
ctys-extractMAClst [options] <dhcp.conf-filename>
```

Description:
ctys-extractMAClst generates a sorted list of 3-column table containing:

<nodename>
<IP-Address>
<MAC-Address>

Alternatively output to stdout could be generated for usage in "/etc/ethers".

<MAC-Address> <nodename>

or

<MAC-Address> <IP-Address>

Therefore this tool requires as source a valid "dhcpd.conf"-Syntax as defined by ISC.

Static configured address mappings are supported only. Dynamic allocated leases of address ranges are not supported.

ctys-tools generally expect for input generated files with default column-order, which is defined by "-n".

Options:

-E
Generates output for "/etc/ethers". This is required

when using ether-wake for WoL, what is no longer supported by ctys.

The options "-n" and "-i" become in combination with "-E" a slightly different semantic:

-n output is "<MAC-addr> <DNS name>"

-i output is "<MAC-addr> <dotted IP address>"

The option is not supported with "-E".

-n|-i|-m

-n Print sorted records: <name>;<MAC>;<IP>

-i Print sorted records: <IP>;<name>;<MAC>

-m Print sorted records: <MAC>;<name>;<IP>

-h

Print help.

-p <db-dir-path>

Directory for data to be stored. This is - and has to be in this version - the same directory as used by ctys-vdbgen and ctys-vhost.

So each file-based ctys-DB requires it's own mapping file for now. This is going to be extended when the LDAP based addressing is introduced. Anyhow, this feature will remain, because it has some advantages for daily business when setting up small to medium networks or multiple test-environments.

The hard-coded filename is "macmap.cfdb"

-P

Almost same as "-p", but takes default ctys-file-DB, provided by DEFAULT_DBPATHLST.

Default output is stdout.

-V

Version.

-X

Terse output format, effects "-V" when set left-of.

Arguments:

<dhcp.conf-filename>
This tool requires a valid "dhcpd.conf"-Syntax.

Exit Values:

- 0: OK
  Result is valid.
- 1: NOK:
  Erroneous parameters.
- 2: NOK:
  Missing an environment element like files or databases.

18.4 ctys-genmconf

Usage:

```
ctys-genmconf <options> [<execution-target-list>]
```

Description:
ctys-genmconf generates the initial configuration entry for a PM or VM, which is stored as

"/etc/ctys.d/[pv]m.conf".

The output is either written to stdout only or duplicated into the previously mentioned file.

The content of this file is required by the VMSTACK feature (refer to Section 13.3.4 'Stacks as Vertical-Subgroups' on page 169) during validation of static consistency and dynamic applicability of the stack members. This is particularly required for hidden files in case of nested VMs to be started, where this data is required to be cached for pre-access within cacheDB. Several additional approaches are available, but this is the most versatile concept, even though it requires some pre-caching efforts.

The current version supports one context only, thus for each change of the booted kernel the differences could require a new generation of the configuration file, including the update of the cacheDB. Future versions are going to support multiple boot-contexts for each PM and VM, which will include the decision for booting the appropriate kernel as decided by the foreseen load-balancing mechanism.

The utility could be performed locally or remotely by full support of remote ctys-addressing.

The generation of data requires root access for some tools. Namely the utility "dmiencode", which is used to evaluate the UUID of the machine requires for execution root-permissions. Anyhow, some older machines may not have a readable UUID at all.

The utility should therefore be executed once on each participating PM and VM by the administrator during installation, and should be completed manually with additional data.

Preferably the rpm package should be installed during initial installation phase, which will generate the appropriate configuration entries.

The data from the pm.conf will be required for the ctys-vhost utility and is therefore fetched by the ctys-vdbgen utility and is evaluated by ctys-dnsutil for dynamic decision of node type.

Options:

-h
Show help.

-I <TCP-address>
Preset value.

-k <ctys-vhost-search-key> The search key to be used as filter for match-only results from the "macmap.fdb" avoiding the usage of local parameters for "host, TCP/IP, and OS". Any valid argument for "ctys-vhost" is applicable.

-M <MAC-address>
Preset value.

-P
Generates standard file path '/etc/ctys.d/[pv]m.conf'.

Default output is stdout.

-r
The range of interfaces to be included, current version supports ethernet interfaces only with the following ranges:

WITHIP
Requires an IP address, which is the "inet" line, currently for IPv4 only.

This is current default.

WITHMAC
Requires a MAC address. An IP address is not neccessary, but could be present.

ALL
Enumerates all ethernet interfaces which is "WITHIP||WITHMAC", localhost is excluded.

-u
Generate a UUID.

-U <UUID>
Preset value.

-V
Show version.

-x <category>
Configuration files for the various machine categories.

PM
Generates the file "/etc/ctys.d/pm.conf".

VM
Generates the file "/etc/ctys.d/vm.conf".

The output is slightly different.

Arguments:

<execution-target-list>
An optional list of <execution-target>. When the "-P" option is choosen, the remote files will be updated, when suffitient permissions are available, else the output is collected locally. The call is simply mapped to a call of the CLI plugin with the option CMD , thus works syncronuous and sequential.

Exit Values:

- 0: OK
  Result is valid.
- 1: NOK:
  Erroneous parameters.

- 2: NOK:
  Missing an environment element like files or databases.

18.5 ctys-groups

Usage:

```
ctys-groups [options]
```

Description:
ctys-groups lists groups from groups DB and groups caches. This tool is just a convenience wrapper for ctys-vhost, which is the interface for network related data storage.

Options:

-c
: List cached groups the output format is:
`<filesize> <lines=members> <group-cache-file-path>`

-d <level>
: Debug.

-h
: Print help.

-l
: Lists group definitions.
`<filesize> <lines=members> <group-name>`

-m <1|2|3>
: List group members in different formats.

-V <version>
: Version.

-X <terse>
: Terse.

Arguments:

<group-list>
: An optional list of groups to be displayed only, if not provided all present groups from directories within the variable CTYS_GROUPS_PATH are scanned and displayed. For additional information refer to the "-S"

option of ctys-vhost .

The format of <group-list> is:

```
<group-list> =: <group-name>[(%| )<group-list>]
```

The seperator is here slightly different from the "ctys-vhost" option. For ctys-groups optionally the common suboptions-argument seperator "%" or the common UNIX arguments seperator "SPACE" could be used as seperator.

The nested containment hierarchy by "include" is expanded before output in any case.

Exit Values:

- 0: OK
  Result is valid.
- 1: NOK:
  Erroneous parameters.
- 2: NOK:
  Missing an environment element like files or databases.

18.6 ctys-macros

Usage:

```
ctys-macros [options] <seach-string>
```

Description:
ctys-macros searches within a macro file and lists the matched macros as requested. Therefore the standard path is searched for a macro file with the provided name or "default".

The result could be filtered by adding an optional filter rule as argument.

Options:

-a
: Adds atoms to the displayed list.

-c
: Adds combined to the displayed list.

-d <level>
: Standard debug, sparsely used here.

-D
: List non-expanded definitions, default is names only. The scope of display (atoms, combined) is not influenced.

-E
: List expanded definitions. Sets implicitly the "-c" option.

-f <macro-file>
: The basename of the macro file to be searched within the standard path list.

  DEFAULT:"default"

-h|–help|-help
: Print help.

-l
Lists available files within the standard searchpath. Each of this has to be fully self-contained file, which could be selected by usage of it's basename only. The selection order in case of multiple occurance is "first-wins".

-V <version>
Version.

-X <terse>
Terse.

Arguments:
None.
Exit Values:

- 0: OK
  Result is valid.
- 1: NOK:
  Erroneous parameters.
- 2: NOK:
  Missing an environment element like files or databases.

18.7 ctys-macmap

Usage:

```
ctys-macmap [options] <seach-string>
```

Description:
ctys-macmap searches within the given list of "macmap.fdb" for matching lines, and shows the requested fields of result. Simple awk-regexpr as for "ctys-vhost" are supported.

The result is pre-sorted, though in case of multiple results these will be presented sorted.

The contents of "macmap.fdb" are given as one record on each line with the following fields.
<nodename>;<IP-Address>;<MAC-Address>
For the generation of macmap.fdb from standard dhcp.conf and the limitation to static assignment data refer to "ctys-extractMAClst".

Options:

-n|-i|-m
-n Print: <name>
-i Print: <IP>
-m Print: <MAC>

-h
Print help.
-p <db-dir-path-list>
Two syntaxes are supported for <db-dir-path-list>:

PATH-like with FS=':':
<db-dir-path-list>=<db-dir-path>[:<db-dir-path>[:...]]
ctys-conformant with FS='%', due to leveled scanner-hierarchies:

<db-dir-path-list>=<db-dir-path>[%<db-dir-path>[%...]]

Both supported field separators are reserved values.

Directory paths for databases, containing a file with hard-coded name "macmap.fdb".

When missing, the default DEFAULT_DBPATHLST will be used.

ATTENTION: Currently no spaces are supported within pathnames!

Arguments:

<simple awk-regexpr>
A "simple" regexpr for awk, used as in given extract:

```
awk -v s="\${argLst}"
  '\$0~s\&\&n==1  '{cache=cache \$1;  mx=1;} ...'
```

Therefore e.g.

```
ctys-macmap  -n -m '00:50:56:.3:..:.9'
```

generates the following matches:

```
tst009;00:50:56:13:11:39
tst108;00:50:56:13:11:49
```

EXAMPLES:
The following examples describe some variants of starting a remote machine by Wake-On-LAN. This is utilized within ctys when using the PM plugin "LINUX".

The evaluation of the MAC-address could be done by several keys from the "macmap.fdb" database. This is the common source when using

```
"ctys-macmap -m <execution-instance>"
"ether-wake <mac-address-from-ctys-macmap>"
```

or

```
"ctys -t LINUX -a create=<-WoL-target> <execution-instance>"
```

ctys-macmap is particularly useful, when the map-database is not yet generated and e.g. local MAC-address for Wake-On-LAN is required for a known name of a PM. Than just call:

```
"ctys-macmap -m <pm-name>"
```

getting the MAC-address

"00:50:56:13:11:39"

For Wake-On-LAN a.k.a. WoL use the following e.g. in combination with ksu of Kerberos:

```
"ksu -e ether-wake ‘ctys-macmap -m ws1‘"
```

For using "ksu" as a sudo replacement edit "/root/.k5users" only, "/root/.k5users" is not required.

For further information on "ksu" as a replacement for su/sudo refer to Kerberos manuals.

For further information on activating "wol" refer to "http://ahh.sourceforge.net

Todo:

Introduce "vendor.fdb" for MAC-Vendor-replacement.

Exit Values:

- 0: OK
  Result is valid.
- 1: NOK:
  Erroneous parameters.
- 2: NOK:
  Missing an environment element like files or databases.

18.8 ctys-plugins

Usage:

```
ctys-plugins [<options>] [<execution-target-list>]
```

Description:
"ctys-plugins" is a versatile test-tool for verification of current status of installation. Due to some priority based restrictions it operates partly on a slightly basic level, involving some specific pre-defined trace levels for output control. Anyhow, the required options for quick-usage are documented in the following sections. Extended trace-levels neccessarily require some deeper familiarity with internal design quickly, which is normally nit required for top-level problem handling.

The utility could be performed locally or remotely by full support of remote ctys-addressing without context options on the command line. If for whatever reasons remote context options are reuqired, than could be applied to macro definitions only.

For now the following main tasks are performed by ctys-plugin:

- display of operational states and capabilities of plugins
- distinguish the check-scope of verified features by "-E" for client-site call-RELAY and server-site final-EXECUTION
- display of called system tools with the actual resolved path-prefix
- display of the actual access permissions as configured by "/etc/sudoers" and/or "$HOME/.k5users"/"$HOME/.k5login".
- display of any level of system initialization traces during call bootstrap, refer to common ctys-standard "-d" option

The output is splitted into 3 sections:

1. The first section is the bootstarp of the tool itself, this includes the basic intialization of the framework, where only framework specific options are evaluated. Typical for this is the "-d" options, which is prefetched by the library itself.
2. The second section contains the trace output during initialisation of the plugins. The content dependes on the choosen debug level. Error messages are displayed in any case.

```
Checking PLUGINS-STATEs now...
------------------------------
<trace-output>
------------------------------
...results to:
```

3. The third section contains the processed results of the evaluated raw component states. Therefore an basic overview of the hosting system is given, followed by a sum-up of the variuos sets of plugings related to the different operational states. The final list contains the operational details for each inidividual plugin.

Options:

-d <common-debug-option>
This option is the common analysis and debugging facility of the UnifiedSessionsManager. Due to the wide scope it is maybe somewhat "like a developer interface".

Anyhow, the most important application for a user during installation and first time systems configuration is the analysis of the called system utilities. This frequently leads to some trouble, which might be obvious, once the following call is executed:

```
"ctys-plugins -d 64,P -T all -E"
```

This call checks all system calls for the current node as a final execution location. The availability as well as (some) access permissions are evaluated. When things don't work, this call is the first instance to "ask".

The "-d" option activates a bit-pattern style debug-level by "P" suboption. Than the bit "64" is set, which is the predefined variable "D_SYS", tracing the internal call-wrapper, almost exclusively used for system calls. When not appropriate, the workaround is implemented in a neatless style.

The "-T" option sets simply "all" plugings to be loaded and initialized.

The "-E" option executes a final destination call, instead of an initial or intermediary RELAY-call.

The following call performs almost the same, but as a client or RELAY.

```
"ctys-plugins -d 64,P -T all"
```

-E

Check for local host as final execution target, this forces full verification. If "-E" is not set, only required functionality for a client role is validated, which could be for some packages almost the same and though treated as.

- This is e.g. the case for PM.
- This is e.g. not the case for XEN, which obviously requires a completly different runtime environment for it's clients than within the server as Dom0/DomU
- This e.g. could be or not the case for VMW, depends on type of product + version + requested type of client.

-t <plugin-type>

The type of plugin to be set to BUSY(4), this is any SINGLE plugin as applicable by "ctys -t ..." call.

-T <plugin-preload-typelist>

The prefetch list of plugins to be set to IDLE(2), before performing, this is any comma seperated LIST of plugins as applicable by "ctys -T ..." call.

Arguments:

<execution-target-list>
An optional list of <execution-target>. When the "-P" option is choosen, the remote files will be updated, when suffitient permissions are available, else the output is collected locally. The call is simply mapped to a call of the CLI plugin with the option CMD , thus works syncronuous and sequential.

Exit Values:

- 0: OK
  Result is valid.
- 1: NOK:
  Erroneous parameters.
- 2: NOK:
  Missing an environment element like files or databases.

18.9 ctys-setupVDE

Usage:

```
ctys-setupVDE [options] [<action-arguments>]  \
       [<execution-target-list-arguments>]
```

Description:
ctys-setupVDE encapsulates and combines a subset of functionality for required tools supporting the creation of devices for network access by QEMU.

The utility could be performed locally or remotely by full support of remote ctys-addressing, including context specific target-options, MACROS and GROUPS. E.g. the required system permissions could be preconfigured for specific users by "ksu" and/or "sudo", for additional information refer to Section 30.1.5 'VDE Remote Configuration' on page 586

ATTENTION:

The remote-execution includes some inherent pit-falls to be considered thoroughly!

This is the case, when this utility has to be executed on a remote site, where not yet a bridge (the only supported networking device for now) exists. During the creation of the required bridge - the so called "main virtual-bridge" (see figure: 8.9), the network will be disconnected for a short time, so any access to NFS or any other networked file system will be interrupted temporarily, which leads to eventual missing of additional tools to be called, e.g. for reconnecting. The same is true for authentication, when kerberos based "ksu" or "sudo", or any other network centric authentication is used in a non-cached environment, so for non-root users the access to system resources is frequently rejected. Particularly the reconnection of the

network device.

Thus remote execution is not approved for users with a mounted remote-home, even though it might work under specific conditions. Local-only users with "sudo" control by complete locally configured environments are verified to work stable.

A specific behaviour of the current version is applied to the created main-bridges. These will get the same IP and MAC addresses as the logical interface, anyhow it works perfectly, as long as you can cope with multiple interfaces with same address information within applied tools. For the functionality of the UnifiedSessionsManager this is handled by a "sort -u" on resulting enumeration IF-lists.

One reason for "doing" the bridge allocation this way is the minimized risk of detaching the remotely handeled VMs for too long from the network services, which might make them unusable from than on. This concept will be probably modified in future versions.

Anyhow, the remote usage of "ctys-setupVDE", once the authentication is configured properly and security facilities are setup thoroughly, offers a tremendous functionality to centralized setup of VM stacks. This is particuarly true in combination of remote usage of ctys-genmconf ans ctys-plugins .

The usage of ctys-setupVDE assures the appropriate environment for the used of the wrappers "vdeq" and "vde-qemu" of the package VDE[130, sourceforgeVde], which is the recommended tool when TAP devices for Linux has to be created. Anyhow, this utility could be used in any comparable case too, but fit particularly for QEMU setup.

The same configuration as for starting QEMU is used. Therefore neatless communications by usage of QEMU-SOCK is guaranteed. The variable QEMUSOCK is based on the variable CTYS_SOCKBASE, which is the default

base directory, where UNIX domain sockets are created. This should be used for eventual additional UNIX domain sockets, such as tcp based serial ports or monitoring devices, too. For additional applicability refer to the user manual of QEMU and to the templates provided by UnifiedSessionsManager.

The following tools are bound together within a script:

- vde_tunctl
- vde_switch
- unixterm
- nc
- brctl
- ifconfig
- /etc/init.d/network

Two types of virtual bridges/switches(see figure: 8.9) are managed by ctys-setupVDE

- "main virtual-bridge"
  The switch to be used for interconnecting the "external" interface, which is in case of the hosting machine itself a physical NIC.
  This switch is created if not yet present, but has to be deleted manually by the user.

- "vde_switch"
  The switch to be used for attachment of VMs. This switch is completely managed by ctys-setupVDE.

ctys-setupVDE prepares a TAP device and with an attached new bridge, therefore it requires the *"Virtual Distributed Ethernet" - VDE(vde2)*[130, sourceforgeVde] package. Additional information with a WiKi containing some very helpful tutorials [132, Basic Networking] could be found at *"VirtualSquare"*[131, VirtualSquare].

In current implementation some assumptions are made in order to ease design and implementation. Anyhow, for

practical application these constraints might not be an important matter.

- one TAP for each vde_switch
- each user has one switch which communicates by default via "/var/tmp/vde_switch0.$USER".
- the management interface for each switch is by default "/var/tmp/vde_mgmt0.$USER".
- appropriate access permissions are provided by sudo or ksu, for automatic detection the ctys framework is used

The following steps are performed by ctys-setupVDE:

1. Creation of a TAP device.

   "vde_tunctl -u <user-without-root-permission>"
   e.g.

   vde_tunctl -u acue
   Returns a line like:

   "Set 'tap3' persistent and owned by uid 4711"
2. Use the returned 'tapX' for networking.

```
ifconfig $1 0.0.0.0 up
brctl addif $2 $1
```

   Does the same as:

   /etc/xen/qemu-ifup tap3 xenbr0

   Which brings up the newly created interface 'tap3' and adds an interface to the virtual Xen bridge connecting it to the world outside.

   The results could be verified with:
   - ifconfig tap3
     should list an interface 'tap3'

- brctl show
  should contain an interface 'tap3'

3. Connect the device.
   Now this interface will be connected to another virtual switch, the vde_switch in order to provide an internal multiplexer for multiple QEMU instances to be connected to the external interfaces e.g. via a present Xen-bridge.

```
QEMUSOCK=/var/tmp/vde_switch0.$USER
QEMUMGMT=/var/tmp/vde_mgmt0.$USER

vde_switch -d \
           -tap tap3 \
           -s ${QEMUSOCK} \
           -M ${QEMUMGMT}


chown -R <userX.groupX> ${QEMUSOCK}
chown -R <userX.groupX> ${QEMUMGMT}
```

   The state could be veriefied with:

```
QEMUMGMT=/var/tmp/vde_mgmt0.$USER

unixterm ${QEMUMGMT}
```

For additional information refer to "Examples" for the Section 30.1 'TAP/TUN by VDE' on page 579 .

Options:
All options are "optional".

-b <virtual-bridge>
The virtual bridge connected to the external network to be attached by TAP device. Default is to use the first bridge detected by brctl. If none is present, tha by default a new one is created with the name "ctysbr0", and the first found interface is added to the bridge.

When an interface is provided by "-i" option and a new bridge has to be created, this will be used instead of

the first valid.

-d <level>
Sets debug.

-f
Forces execution even when processing seems to be critical.

1. Forces call of "kill <PID>", when here-script with "unixterm ... shutdown" fails.

   For current version this seems to be frequently the case on i386 architecture, whereas x86_64 works.
2. Creates a new bridge, even when connected via a network session. This could interrupt the current calling session permanently, even lead to it's hang-up due to a required short-time disconnect. So this should preferrably proceeded from within a local session.

-g <sbit-group>
Sets the s-bit for the group, this has to be the same as the resulting owner's group.
If not set, the resulting permissions for QEMUSOCK are "rwx——", else "rwx–S—".

-i <interface>
The interface to be added to a newly created bridge, see "-b" option.

-s <ALTERNATE-QEMUSOCK>
A file-socket to be used for communications peer via virtual switch. Default is set by common QEMUSOCK configuration.

-S <ALTERNATE-QEMUMGMT>
A file-socket to be used for management console of virtual switch. Default is set by common QEMUMGMT configuration.
Could be used with "unixterm $QEMUMGMT" of *VDE(vde2)*[130, sourceforgeVde].

-u <non-privileged-user>[.<group>]
Owner of the created TAP device. Default is current user.

-h
Print help.

-V
Version.

-X
See ctys, terse for machine output.

Arguments:

cancel
Removes the switch and it's attached TAP device. In case of partial present resources these will be cleared as present, thus remaining parts of partly execution could be reset.

check
Performs basic check for the accesibility of the virtual switch setup for selected USER. Therefore a simple "ctys-setupVDE PORTS" call is analysed for the occurance of at least one "tap" device and one UNIX-Domain socket, which are verified by their existence. In case of erroneous state basic information for further analysis is displayed. Anyhow, still malfunction could occur, but if check fails, it will definetly.

create
Creates a new virtual switch, this comprises a new TAP device and an attached virtual switch. When no bridge is present a virtual bridge is created too, and the tap-device is attached.

The CREATE call just checks whether a functional switch is already present, if not it just creates a new one. Therefore the current defined socket for the management interface is utilized. Thus a new call on a

present, but erroneous switch leads to reuse of the sockets, but creates a new tap-device and starts a new instance of a vde-switch-process. Present tap-devices are not reused, and just kept untouched.

info
Shows vde_switch information.

This is the default behaviour.

ports
Lists ports of vde_switch.

list
Lists present vde_switch-es. The base-switch entries are displayed only.

listall
Lists present vde_switch-es. Any entry is displayed, this includes the dynamic created port specific sockets.

Due to some minor difficulties for now these are not removed, when the client disappears, thus "listall" could be used to check the dengling entries from time to time.

<execution-target-list>
Execution targets to be listed.

Exit Values:

- 0: OK
  Result is valid.
- 1: NOK:
  Erroneous parameters.
- 2: NOK:
  Missing an environment element like files or databases.

18.10 ctys-smbutil

Usage:

```
ctys-smbutil [options]
```

Description:
ctys-dnsutil displays a short list of current available SMB server and workstations.
Options:

-d <debug-level>
Same as ctys.

-h
Print help.

-i
Show numerical TCP/IP-Address. This is supported in combination with "-n" option only without "-X" option.

-n
Show TCP/IP-Address as name. This is supported in combination with "-i" option only without "-X" option.

-R <runtime states>
Restricts a set of multiple results with additional constrains for output.

Only the possible targets which are operable or actually operational are listed. This includes the actual running VM with it's hosting PM, and in addition all other operational machines, where the current VM is available too. This case is the most common for NFS based exec-pools, where a single VM could be accessed remotely by a number of PMs. This particularly offers the advantage of copyless-migration of online and offline VMs.

Very handy, and in companion with others probably one of the most important internal top-level-calls for

GuestOS-Command-Execution.

<runtime states>=[(REVERSE|R|-),]PING|SSH[,PM|VM]

REVERSE|R|- This reverses the resulting set, thus the "not matching" states only will be shown.

PING[:<packetcnt>[%<timeout>]] A RUNNING session is simply "ping-ed".

Resulting statement of operational mode may result on applied security policies, but not exclusively on the state of the running OS's IP-stack.

-V
See ctys, version output.

-X
See ctys, terse for machine output.

Exit Values:

- 0: OK
  Result is valid.
- 1: NOK:
  Erroneous parameters.
- 2: NOK:
  Missing an environment element like files or databases.

18.11 ctys-vdbgen

Usage:

```
ctys-vdbgen [options] <target-list>
```

Description:
ctys-vdbgen generates databases of static mapping from configuration files for usage by ctys-gettarget. This tool itself is a wrapper for

```
"ctys -a ENUMERATE=machine...."
```

The output of the ENUMERATE action for a given list is stored into one file contained in the given DB-PATH, or default respectively.

The mapping contains available VMs for a given list of PMs.

Several files could be generated as specific access-groups which could be selected by parameter for ctys call.

The content will be used as runtime-decision base for additional selective evaluation by "ctys -a LIST=MACHINE" call.

REMARKS:

1. Multiple-Entries of IP, and/or MAC addresses
   All entries are evaluated and checked for matching indexes. This is for an "IP0" an "eth0" is searched, if missing a warning is generated. Whereas "eth0" without an IP address will be accepted.

The second specific is the generation of one seperate entry for each of the resolved IP-MAC pair.

2. Other Multiple-Entries
   The common behaviour in case of multiple entries is to use the FIRST-ONLY. Even though some tools might present more than one, it must not be relied on!
3. Whitespaces
   Whitespaces are generally no supported. When required within suboptions the '%' sign has to be used for padding field-separators.

   Anyhow, please avoid them, at least for now!

Due to the defaults described in the following subsection for options, the call

```
ctys-vdbgen <host1> <host2> ...
```

leads to the default call

```
${HOME}/bin/ctys -a enumerate=machine,b:\$HOME >\
  ${HOME}/.ctys/db/default/enum.fdb <host1> <host2> ...
```

Where the PM plugin by default additionaly checks for the PM configuration which is located in

```
/etc/ctys.d/pm.conf
```

For additional information refer to "ctys-genmconf" .

A special append-mode is supported for addition of data. This mode does a pure concatenation only, not redundancy of added data is tested. Therefore the user should be aware, when calling append mode, whether he already updated the data of that node.

The deletion of data has to be performed manually for now, a simple ASC-II editor, MS-Excel, or the spreadsheet application of OpenOffice could be used for this task.

The data format is described in the appendix of the ctys manual.

Options:

ATTENTION:
The following options are case sensitive, thus are required literally. If not printed properly, word-options have 2 leading hyphens '-'+'-'="–".

<same as "ctys -a enumerate=machine,b:<srcpathlst>"
With specific environment variables for modification of the default search path DEFAULT_ENUM_BASE.

```
<srcpathlst>=
   "\${BASEPATHLST:-\$DEFAULT\_ENUM\_BASE}\
\${BASEAPPENDLST:+\%\$BASEAPPENDLST}"
```

which expands by default to

```
"ctys -a enumerate=machine,b:\$DEFAULT\_ENUM\_BASE ..."
```

All context options and suboptions will be passed through, so apply same as for ENUMERATE action.

–append
This activates the append mode, just appending data to present fdb or creating a new if missing. The modes append, replace, and stdio are exclusive and could not be combined.

This is foreseen for addition of minor changes and avoiding a long running recollection of each data set from each target.

-b
Background mode, refer to common options "-b" .

–base=<base-path-list>
A <base-path-list> to be scanned exclusively. This can be combined with multiple targets and the "–append" mode for quick addition of specific new VMs. The path could be the contianing directory for a single VM only.

-c
Nameservice caching mode, refer to common options "-c" .

-C
Result data caching mode, refer to common options "-C" .

–cacheDB=<output-db-directory-path>
Pathname for directory containing DB file to be created. This file stores the mapping records generated from ctys-ENUMERATE literally.

The evaluation order priority and predefined default values for the directory is defined as follows:

1. "–cacheDB=<output-db-directory-path>"
2. DEFAULT_VDBGEN_DB=$HOME/.ctys/db/default
3. DEFAULT_DBPATHLST=$HOME/.ctys/db/default

or

1. "–cacheDB=<output-db-directory-path>"
2. DEFAULT_VDBGEN_DB=<install-path>/conf/db/default
3. DEFAULT_DBPATHLST=<install-path>/conf/db/default

The name for the filedb itself is hard-coded as "enum.fdb". Additional files are be stored within the cacheDB directory.

REMARK:
Only ONE path could be provided here for the PATHLST.

–filecontext
This switches off several defaults and assumes that a configuration file with a complete CONTEXT is provided.

The explicit suppression of defaults is required, due to the addition of chained context options for dialogue based entries from right to left. The evaluation of superposing options is proceeded from-left-to-right. Thus the last wins, so the last dialogue entries have the highest priority.

–progress
This activates the formatted table output, therefore some traces are filtered and displaye in a compact overview,

indicating the progress and required processing time.

For the application of the "–progress" option additionally the appropriate trace switches for the target has to be set. This is due to the required activation of the appropriate trace output for post-filtering on local machine. In case of usage of MACROS or GROUPS these could be stored permanenetly as required.

E.g. the following debugging("-d") context options for the target of enumeration required to be present.

```
ctys-vdbgen \
  --cacheDB=/home/tstusr/.ctys/db/tmp \
  --append \
  --base=qemu \
  lab00'(-d 2,s:16,w:0,p)'
```

When "progress" choosen the data for each displayed instance is a post-result, calculated as a sumup after finished processing. For continuous display refer to "progressall".

–progressall
: This activates the formatted table output with continuous display, where the data is filtered as well, but any relevant entity with an intermidiate result is displayed. The final sum-up data is additionally displayed as soon as the somplete set of results for the current entity is available.

–replace
: This activates the explicit replace mode, which is required when a "enum.fdb" alread exists. Choosing this will delete the present "enum.fdb", when the data has to be kept, make a backup before activating this option. The modes append, replace, and stdio are exclusive and could not be combined.

–stdio

Writes it's output to STDOUT only. The modes ap-

pend, replace, and stdio are exclusive and could not be combined.

-t
Session type, refer to common options "-t" .

-T
Preload of session type plugins, refer to common options "-T" .

Arguments:

<target-list>
Any target to be enumerated.

Exit Values:

- 0: OK
  Result is valid.
- 1: NOK:
  Erroneous parameters.
- 2: NOK:
  Missing an environment element like files or databases.

18.12 ctys-vhost

Usage:

```
ctys-vhost [options] <awk-regexpr>[ <awk-regexpr>[ <...>]]


ATTENTION: Default behaviour is no output, thus
you need to select one, e.g. "-o ...".
```

Description:
ctys-vhost is the basic address resolution interface for runtime execution of commands based on ctys addressing.

The similarity of UNIX "host" function is expanded with several features, which take into account the roaming of VMs and thus changing their actual execution path within a so called "execution stack" assembled by PMs, VMs, and HOSTs.

Due to technical reasons the locator functionality required for building complete fully qualified ctys-addresses of execution stacks, which could be seen as a UNIX "ping" similarity, is included.

Particularly a basic load-balancing is included, which is very basic of course, but could be extended easily. The current version requires at least one of the potential Exec-Targets to be active, which could be a PM for execution of a VM to perform a command, or a VM for execution of a command only, no automatic start of deactivated sessions are performed.

The main task of this tool is supporting a scripting-IF as a link with an convenient name-binding scheme between a GuestOS and it's containing VM and PM by an open and GuestOS-Native interface. The whole access and security facility of VMs and PMs including HOSTs(e.g. VNC) is handled by encrypted connections only. It is designed and

implemented as a seamless SSH-based authorization and authentication system. The authors environment utilizes Kerberos, LDAP, and SSH with automount for SSO. The ctys-vhost supports mainly the glue for seamless binding of roaming VMs on a homogeneous UNIX platform.

Simplicity and common extendibility by widely used bash is another present feature.

Due to targeted simplicity and efficiency the address resolution is based on simple unstructured pattern matching by regular expressions onyl whenever possible(which is almost for each call). Even though the line-record is structured by fields as given by "-a LIST=MACHINE" and "-a ENUMERATE=MACHINE", the match itself will be performed as a simple regexpr by "awk-match". Whereas the output is performed on the level of fields. Multiple regexprs are supported and will be iterative applied on the intermediary results. The reduction of the output as requested by the "-o" option is performed on the final set of resulting records only.

This fits perfectly, as long as the given IDs are kept unique, when ambiguity occurs, the match will be taken as defined by "-M" option.

REMARK:

This flat-and-simple approach works as good or better to say much better than any "Attribute-Value-Assertion" or any other kind of an Object-Tree based access. And the best is, you can implement it within some days, without any framework to be used!!!

But yes, some selection scope could be tricky, anyhow, for almost any practical relevant query it works fine. As could be seen.

Ambiguity occurs frequently when using a VM with NFS(or an other some more secure network file system) from multiple machines and using the nodes as processing-capacity-only. Accessing the same files and enumerating them for

selection of the appropriate execution machine could be a mandatory requirement for load-distribution policies. In this case a basic COST option "-C" supports the very basic "-M" option for some quite usable load distribution within execution-groups.

In most other cases uniqueness should be given, e.g. the key UUID is defined to be unique, but could be tampered e.g. by co-allocated backups. For avoiding of backup-access the "-M first" option might be helpful.

Anyhow, the management of up to some hundred VMs might not be a challange by the current file-DB and not really performance-optimized toolset. One of the next versions will additionally support LDAP based nameservices, targeting an enterprise environment with "unlimited" and "volatile" distributed services to be managed.

When a MAC-IPAddress-HostName mapping table a.k.a. "macmap-DB" is present this will be used for open mappings which are not configured within the VM configuration files a.k.a. enum-DB. Particularly any IP or PM/Hostname address for given MAC-Address will be resolved when not present within the enum-DB.

The address resolution will be performed by the following steps:

1. Check the static list of given ExecGroup for possible candidates.
2. List the active sessions on the given ExecGroup.
3. Take the appropriate PM/VM by utilising "-C" and/or "-M" option.

For additional help refer to online help within "ctys" to the section "NAMESERVICE-BASE".

Now some real benefits when using ctys-vhost as interactive tool instead of using id from scripts:

- Given partial strings, e.g. "192.168.1" lists all machines of that subnet. When the "-M active" option is choosen, all currently active sessions within that subnet are listed.

- Any string could be used as partial pattern, e.g. parts of MAC-Ids of fragments of UUIDs. The given string will be matched against complete record, mostly an awk-regexpr, thus any part, even spanning multiple FIELDS could be used. But currently not regexpr, just literal characters are supported.
- The database founding the mapping information of ctys-vhost could be altered by "-p" option for handling of multiple sets, e.g. for test-purposes.
- The databases enum-DB and macmap-DB are populated just with the native information provided by their main sources, dhcpd.conf and the config-files of supported VMs. Therefore not any information might be present in each of them, e.g. the IP-Address of the GuestOS might be present within the macmap-DB, but not within the VM-config. The "-S" option allows the combined usage of multiple sources, e.g. by values "all", "macmap", or "enum".
- ctys specific configurations-extensions as described for the "-a ENUMERATE" option are fully supported. This includes particularly the storage of GuestOS information within the VM-config by specific ctys-Prefixes(#@#) and some helpful keywords.

The format of the generated data records is literally the same as the MACHINE output of the ENUMERATE action.

Additionally to the flat-matching by simple regular expressions some additional keywords are defined. The are AND , OR , NOT , E: ,and F: , which are described within the section related to the arguments.

Options:

-c <spent cost on execution environment>
Cost as for load distribution when selecting a target.
<spent cost on execution environment>=MINCNT|MAXCNT

MINCNT Gives minimum loaded target, number of given types are simply counted.

MAXCNT Gives maximum loaded target, number of given types are simply counted.

CNT Lists each target with it's TYPE-COUNT.

Companion options apply to resulting set of equal cost.

-C <DB sources>

Limits the generation of the cache DB to the for mapping-resolution to the listed sources. Default is to use all. Only available databases will be used, missing are silently ignored.

Due to some performance issues when repetitively accessing same temporary runtime data, some internal caches are defined. These can be controlled, and reused or cleared by usage of some of the following keywords. But additionally some automatic checks apply.

For data from static information, which has to be pre-processed a local cache-DB is created. This cache-DB will be checked for modification time of it's sources before each access and updated when outdated.

The modification time of the cache files will be checked additionally for their age. When these exceeds the value defined by CACHECLEARPERIOD, which is by default 3600seconds, the caches are forced-cleared and rebuild silently by next call.

The following data sources are utilized:

ENUM

Enumaration results only, as supplied by cached local"enum.fdb".

MACMAP

DHCP information for MAC resolution, the macmapDB should be available, but is otherwise simply ignored.

This will be utilized in conjunction with an enumeation result, e.g. ENUM.

GROUPS

Adds caching of GROUPS for all group files from the current CTYS_GROUPS_PATH variable. Therefore each group file will be completely expanded by nested evaluation and replacement of "#include" statements and stored by replacing each resulting entry with it's MACHINE format entry from the staticCacheDB.

Each group will be cached within an file by it's own, thus the access could be performed by just one file-selection for the complete nested resolution of it's entities.

```
<DB sources>=
OFF|
CLEARTMP|
CLEARALL|
GROUPS|
KEEPALL|
LIST|
LISTARGETS|
LISTGROUPS|
MEMBERSDB|
MACMAPONLY|
MACMAP|
REBUILDCACHE
```

This group of keywords controls the runtime behaviour, which has an impact to the overall performance.

OFF
: Bypasses the usage of caches.

MACMAPONLY
: Uses the macmap.fdb only for mapping, this is just senseful for mappings between DNS, MAC, and TCP. The request will be rejected, when "-o" option contains any other input.

  For matching entities within MACMAP this might be the fastest approach. It is the only applicable approach, when the target is not yet populated in standard DB, for example due to pre-initial conditions.

MACMAP
: Activates the raw usage of macmap.fdb for DNS, MAC, and TCP as preferred source of resolution.

This has two flavours, depending from selected output attributes:

1. Only one of, or all: TCP|MAC|DNS
   In this case the MACMAP DB will be utilized within the "bigger awk", due complete probable containment of information thus first a raw access to MACMAP will be tried. When no result was found, the general script with DNS/Bind access will be performed. In standard manner(due to SW architecture, ignoring previous trial).

2. Additional output requested:
   In this case particularly the field positions of the resulting output can not be handled in a smart manner for an independent pre-filter, though the standard execution path is performed.

When the macmap.fdb is properly maintained and contains the complete scope of mapping information, this enhances the performance, else it could have an negative impact, even though it will not be dramatic, or for small amounts almost not recognizable.

A second aspect to be aware of is, that the two different databases might diverge. Particularly the order of the stored records could not be relied on to be the same. When using the option "-M all" the order might not be relevant, but for "-M first"(default) and for "-M last" the results might frequently be different.

The basic difference of the contents is the fact, that the macmap.fdb (let us say!) contains any networked host, whereas the standard enum.fdb the registered VMs only, so might be a subset of macmap.fdb.

The correlation of both will be performed, when a cache is build and addressing references are resolved for faster access.

GROUPS
Activates the usage of GROUPS and it's related cache data which is due to performance issues deactivated for now by default.

The following additional keywords control and support the management of internal caches.

LISTCACHE
Lists all current caches.
This call terminates immediately after performing, so any remaining options are ignored.

LISTTARGETS
Lists all current cached targets.

This call terminates immediately after performing, so any remaining options are ignored.

LISTGROUPS
Lists all current cached groups.
This call terminates immediately after performing, so any remaining options are ignored.

MEMBERSDB
Displays a list of all current staticCacheDB members in ctys-stacked-address notation.

CLEARTMP
Clear it's internal temporary caches first and rebuild on demand.

CLEARALL
Clear all it's internal caches first and rebuild on demand.

This includes a directory-wildcard-clear, which includes probably the caches of other tasks, so use it considerably.

This call terminates immediately after performing,

so any remaining options are ignored.

REBUILDCACHE
The static data to be concatenated from static assembly databases, for now the enum-DB and the macmap-DB is cached within a static database and concatenated with the volatile RT data into the RT-CACHE.

The requirement of rebuild for the static data is checked by modification time of it's components, and when required updated silently.

When setting this flag, the data is rebuild in any case.

Additional information is available from description of:

- NAMESERVICES of "ctys -H"
- "ctys -a ENUMERATE...."
- "ctys-extractMAClst"
- "ctys-vdbgen"

-d <debug-level>
Same as ctys.

-h
help

-i <input-list>
Options controlling input content for specific cases.

`<input-list>=[CTYSADDRESS|CTYS]`

CTYSADDRESS|CTYS
A fully qualified address is supported for mapping of one of the given output attributes.

-I <0-9>
Interactive, gives summarised display of progress for main values. The degree of display depends on the choosen level:

- 0
For completeness only, switches the display OFF, same as omitting the option at all.

- 1
  Activates a moderate level with display of basic benchmark data.
- 2
  Activates a more informative level with intermediate QUERY data reduction pattern. This particularly supports the design of multi-key selection queries for perfomance optimization.

  `ctys-vhost <in-out-options> <arg1> <arg2> <agr3>`

For the display of the actual contents of a specific intermediate step in addition to it's draft performance-overview, just drop all following filters/arguments from the call, what will display the requested result as final. This result is identical to the covered intermediate result when using it within a chained set of filters.

-l <USER>
Remote user to be used for SSH-access-checks, when the "-R" option is activated.

`DEFAULT=CTYS\_NETACCOUNT(DEFAULT->USER)`

-M <result-set-output-reduction>
Restricts a set of multiple results with additional constrains for output:

`<result-set-output-reduction>=FIRST|LAST|ALL`

FIRST First matching entity.

LAST Last matching entity.

ALL All matching entities.

COMPLEMENT
All entities NOT matching.

SORT Final result is sorted by "sort".

USORT Final result is sorted by "sort -u". Only full matches are reduced.

UNIQUE Final result is sorted by "sort -u" but only displayed when actually one record only results. When multiple records are matched, an empty string is returned and the exit value is set to "1".

-o <output-list>
Options controlling output content. Values of all given options are listed as one RECORD per line for each matched entity. The keywords are not case sensitive and could be used as a comma-seperated list. Shortcuts are applicable mostly as one-character alternatives as listed.

The default output when this option is not provided is to display a pre-configured table stored as a MACRO in the default-macros file with the name

TAB_CTYS_VHOST_DEFAULT

This table could be customized as required, but should be handeled carefully.

```
<output-list>=
(
  (
    (
      [ARCH][,]
      [CATEGORY|CAT][,]
      [CONTEXTSTRING|CSTRG][,]
      [CPORT|VNCPORT][,]
      [CTYSADDRESS|CTYS][,]
      [CTYSRELEASE][,]
      [DIST][,]
      [DISTREL][,]
      [EXECLOCATION][,]
      [GATEWAY][,]
      [HWCAP][,]
      [HWREQ][,]
      [HYPERREL|HYREL][,]
      [IDS|ID][,]
      [IFNAME][,]
      [LABEL|L][,]
      [MAC|M][,]
      [NETMASK][,]
      [TYPE|STYPE|ST][,]
      [OS][,]
      [OSREL][,]
      [PLATFORM|PFORM][,]
      [PM|HOST][,]
```

```
        [PNAME|P][,]
        [RELAY][,]
        [RELOCCAP][,]
        [SERIALNUMBER|SERNO][,]
        [SERVERACCESS|SPORT|S][,]
        [SSHPORT][,]
        [STACKCAP|SCAP][,]
        [STACKREQ|SREQ][,]
        [TCP|T][,]
        [USERSTRING|USTRG][,]
        [UUID|U][,]
        [VCPU][,]
        [VERSION|VERNO|VER][,]
        [VMSTATE|VSTAT][,]
        [VNCBASE][,]
        [VNCDISPLAY|DISP][,]
        [VRAM][,]
      )
      [TITLE|TITLEIDX|TITLEIDXASC][,]
      [MACHINE|MAXKEY][,]
  )
  | TAB_GEN[:<tab-args>]
)
[IP|DNS][,]
[,SORT[:<sort-args>]]
```

The previous keywords for specific fields set the related bit for output. These will be OR-ed to the resulting output. Thus the MACHINE keyword includes all fields, whether individually set or not.

The format keys IP and DNS change the representation of the IP field.

ARCH
: The architecture presented by the hypervisor to the GuestOS.

CATEGORY|CAT
: The category of the plugin, which could be for now one of: HOSTs, PMs VMs.

CONTEXTSTRING|CSTRG
: A string stored for the use by responsible the plugin.

CTYSADDRESS|CTYS
: A fully qualified address to be used within ctys. This includes the complete address for the whole execution-stack of the destination instance, beginning with hosting PM.

  Whereas almost any other output is just a subset of the generated static database, this value is the result of the assembly of multiple items to a complete address for an unambiguous execution path. The namespace could be the private network or even the global network, when globally unique PM addresses as FQDN are used.

CTYSRELEASE
: The release of ctys used for creation of the VM.

DIST
: Output of distribution installed within VMs guest.

DISTREL
: Release of distribution.

DNS
: Output of TCP/IP address (any valid for the VM). This option supports the name representation as reported by DNS, for the numerical representation refer to IP.

  ATTENTION: Only the first match will be listed when multiple addresses are present for the same entity.

EXECLOCATION
: The location of execution for the VM. Either a keyword, or a list of hosts/groups.

GATEWAY
: The TCP gateway to be used for the current interface, which is for the standard case the one for the whole multihomed node.

HWCAP
: The offered hardware capacity by the VM to the GuestOS.

HWREQ
: The required hardware capacity of the VM from the PM, which could be a lower peer VM within a stack.

HYPERREL|HYREL
: The release of the hypervisor the current VM is created with. E.g. "Xen-3.0-x86_64".

IDS|ID|I
: Output of static ID, which is a pathname for VMs, and a runtime ID for HOSTs. The IDs are (foreseen to be!?) unique within the namespace of their PM or VM. This should be considered when roaming VMs between PMs.

  Following current exceptions apply:

  XEN
  : The value is the configuration path statically unique on local host, common to IDs of other VMs.

    The domain-ID is handled - due to hypervisor architecture and structural and dynamic means of accessibility - similar to an ordinary "UNIX-pid", but not considered within ctys.

  HOST
  : For plugins of type HOST, which are more or less simple processes offering specific services, the "UNIX-ID" is utilized.

    The "UNIX-ID" could consist of several kinds of entries. A common example is VNC, where the entries semantic could be one of:

    - DISPLAY = VNC-port-offset
    - DISPLAY = VNC-port

    - Any of above could be context-specific, and utilized more or less correlated by any other FBP-aware application too. E.g. vncviewer for XEN and WMWare-Workstation 6.

    In addition, for a plugin a ctys specific ID might be defined, e.g. based on "UNIX-PID".

    So, ... it is just an abstract ID, no generic overall-algorithm applicable.

IP
: Output of TCP/IP address. This option supports

the numerical representation, for the DNS name representation refer to DNS.

LABEL|L
Output of LABEL.

MAC|M
Output of MAC address.

ATTENTION: Only the first match will be listed when multiple addresses are present for the same entity.

MACHINE
Complete records matching the <regexpr-list> in terse format for postprocessing.

MAXKEY
The maximum common set of attributes for LIST and ENUMERATE.

NETMASK
The TCP netmask of current interface.

OS|O
Output of OS as configured.

OSREL
Release of OS.

PLATFORM|PFORM
The HW platform provided for the GuestOS.

PM|HOST|H
Output of TCP/IP address of the PM-Physical Machine, which is the hosting machine.

PNAME|P
The same as <ID|I>, this is due to the usage of filepathname of the configuration as an unique ID at least within the namespace of a sigle hosts filesystem.

RELAY
The relay interface, device, virtual bridge, virtual switch, or virtual hub, the VM is interconnected too witin it's PM/lower-stack-peer.

RELOCCAP
The available capacity for relocation of the VM, either to another compatible virtual PM as a stack-entity, or an actual physical PM. The destination

container has to provide the required HWREQ and STACKREQ of the VM, which has to be compatible with the HWCAP and STACKCAP ot the target.

SERIALNUMBER|SERNO
: An arbitrary serial number for the VM stored in the configuration file. This number should be unambigiuos.

SERVERACCESS|SPORT|S
: Server access port for execution of a TCP/IP connect. This is the raw port to be used for server specific admin tools, which is different from user's client access. This port is currently rarely supported, namely not utilized due to security reasons, e.g. in case of XEN.

  The main intention of ctys is to avoid propriatery interfaces as much as possible, and support "bare support tools" only. This interface could only be propriatery. So being honest, 'do not really like that!

SSHPORT
: A list of provided SSH ports on this interface. Currently supported for OpenSSH only.

SORT
: Enables the post-sort filter.

  `<sort-args>=[ALL|EACH][%UNIQUE][%<sort-key>]`

  UNIQUE
  : Activates a pre-final filter for call of "sort -u".

  <sort-key>
  : Defines a sort key as "-k" option for "sort -k <sort-key>".

STACKCAP|SCAP
: The capacity offered by the hypervisor to nested VMs.

STACKREQ|SREQ
: The capacity required by the hypervisor as a nested VM itself.

STYPE|ST
: Output of the session type, either of category VM, PM, or a HOST by it's plugin name.

TAB_GEN:<tab-args>
: Refer to common format for additional information.

TCP|T
The ip address of the VM in stored format.

ATTENTION: In case of multiple interfaces and/or addresses for each address of a so called "multi-homed" machine a sepereate entry is generated, thus it is listed as a seperate host entry.

TITLE
The title for any selected field within the output.

TITLEIDX
The title with the related indexes as required and enumerated for input into the generic table.

TITLEIDXASC
The title with the related indexes as required and enumerated for input into the generic table. In addition the ASC-II values of column indexes for common spreadsheet forms are displayed.

USERSTRING|USTRG
A free editablecustomizable string from the user.

UUID|U
Output of UUID.

VCPU
The number of pre-assigned VCPUs.

VERSION|VERNO|VER
Version of config.

VMSTATE|VSTAT
The configured state of the VM. Current aupported values are: ACTIVE, BACKUP.

VNCBASE
Base port for calculations of ports from display and vice versa. The default is 5900.

VNCDISPLAY|DISP
DISPLAY to be used by XClients, which in case of VNC is already calculated by usage of context-specific PortOffset.

VNCPORT|CPORT
Client access port for execution of a TCP/IP connect. This is the raw port to be used for vncviewer or proprietary clients with their own MuxDemux-dispatcher.

All configured VNC access ports for any VM could be listed as:

```
ctys-vhost -o cport,l -M all '59\[0-9\]\[0-9\]'
```

Where a standard baseport of 5900 is assumed.

VRAM
: The amount of pre-assigned VRAM.

-p <db-directory-path-list>
: Comma seperated path list to directories containing the name-resolution DBs, same for each <db-directory-path> as for ctys-vdbgen.

  ctys-vhost could handle multiple mapping-DBs for virtual concatenation. The advantage of this is the ability of substructuring VMs and PMs into access-groups by ctys-vdbgen and using them in combinations as required during runtime. This offers particularly advantages when performing ctys-vhost for loadbalancing by usage of cost-option "-C".

-r
: Activates the common usage of dynamic runtime data. Without this option only some distinct functions like load-distribution utilize selective calls of runtime-data-evaluation for further restricting their intermediate results. This is e.g. obviously the count of actual executed instances on a PM for the case of cost evaluation on a potential distribution target.

  When runtime data evaluation is activated in general, the "-R" option alplies to any result as a further constraint.

  The usage of runtime data evaluation cost performance of course. This could become dominant, when huge clusters are evaluated, thus should be considered whether really required, and applied to reasonable sets only. But anyhow, when some bigger sets are required by definition, caching of data with different strategies could be applied.

-R <runtime states>
: Restricts a set of multiple results with additional constrains for output.

Only the possible targets which are actually operational are listed. This includes the actual running VM with it's hosting PM, and in addition all other operational machines, where the current VM is available too. This case is the most common for NFS based exec-pools, where a single VM could be accessed remotely by a number of PMs. This particularly offers the advantage of copyless-migration of online and offline VMs.

Very handy, and in companion with others probably one of the most important internal top-level-calls for GuestOS-Command-Execution.

```
<runtime states>=
[MARK|(REVERSE|R|-),]PING|SSH[,PM|VM]
```

MARK
: A match for any of the following keywords is simply made with a prefix as running by "R;", instead of just showing the resulting set.

  The remaining will be formatted with "-;" as prefix for alignment.

REVERSE|R|-
: This reverses the resulting set, thus the "not matching" states only will be shown.

PING
: A RUNNING session is simply "ping-ed".

  Resulting statement of operational mode may result on applied security policies, but not exclusively on the state of the running OS's IP-stack.

SSH
: A RUNNING session is additionally checked for SSH-connect by a dummy-interconnect.

  This might be really senseful, once SSO is established.

  "ssh" is the only and one state, which is a viable confirmation for the ability of establishing ctys connections.

PM
: Checks only PM for accesibility, which is the default behaviour.

  PM accessibility is defined as the accessibility of the running OS on PM.

VM
: Checks VM for accesibility, this is particularly related to the SSH key.

  VM accessibility is defined as the accessibility of the running OS on VM.

-s
: Set when ctys-vhost is used as an internal subcall for another master-tool. In this case some automatic triggered tasks such as the time-driven rebuild of caches are suppressed. Instead a hint for required re-sync is printed as warning.
  Urgent tasks will be worked out, even if they might take some minutes. This is the case when no cache is present, of the caches differ in their age.
  All tools using this as an internal system call should set this flag.

-S <BasicDataManagementSupport>
: The "-W" option represents some basic management interfaces for the additional entity class GROUPS and the entity characteristics CONTAINMENT. Where the containment is applied to the whole set of stored entities.

  These interfaces allow some smart listing and display of current supported data, the handling of data as deletion and creation is handeled by the ctys-vhost command as appropriate.

```
<BasicManagementSupport>=
LISTALL|
LIST|
LISTDB|
MEMBERSDB|
```

```
LISTGROUP|
MEMBERSGROUP[23]
```

The following keywords may be applied.

LISTALL
: Displays a list of all current available data sources.

LIST
: Displays a list of all current data sources, the same as

  LIST = LISTDB + LISTGROUP

LISTDB
: Displays a list of current file-databases.

MEMBERSDB
: Displays a list of all current staticCacheDB members in ctys-stacked-address notation.

LISTGROUP
: List all current groups from the CTYS_GROUPS_PATH.

  The output format is as follows:

  " <size> <#lrec>/<#incs> <#srec> <group>"

  <**size**> Size n kBytes.

  <#**lrecs**> The overall number of target entities without resolution of nesting, so just the current file is evaluated.

  <#**incs**> The overall number of include-statements contained within current file.

  <#**srecs**> The overall number of target entities with resolution of all nested includes.

  <**group**> The name of current group, which is the filename too. When "-X" option is set (LEFT of this option), than the basename is shown only, else the full filepathname.

MEMBERSGROUP[:<group-list>]
: Lists members of scanned groups. When no <group-list> is provided, the variable CTYS_GROUPS_PATH is decomposed and similar to the PATH variable, any resulting directory is scanned for all existing group files. The members of found groups are displayed.

The nested containment hierarchy by "include" is expanded before output.

In case of provided <group-list> the listed groups are displayed only. The format of <group-list> is:

```
<group-list> =: <group-name>[%<group-list>]
```

Two types of storage are shown:

- Raw group files, which may contain target entities, include-statements and comment lines.
- Cache group files, which contain the whole resolved set of containment tree as flat target entity recorded from the statCacheDB.

(MEMBERSGROUP2|MEMBERSGROUP3)[:<group-list>]
Same as MEMBERSGROUP, but with slightly different output format.

Additional information is available from description of:

- NAMESERVICES of "ctys -H"
- "ctys -a ENUMERATE...."
- "ctys-extractMAClst"
- "ctys-vdbgen"

-T <type-list>
Types to be recognized when calculating target. For additional information refer to "-T" option of ctys.

-V
Shows version.

-X
See ctys, terse for machine output.

REMARK: Due to order dependency of options evaluation, set this as first/leftmost option.

Arguments:
It is recommended to try the "-I 2" option for some performance analysis of order dependency for multiple-selection queries.

<awk-regexpr>[<awk-regexpr>[<...>]]
A list of simple awk regular expression, for matching based on $0. This is called here "flat-matching", though no structural information like in case of attribute-value assertion, is recognized for the pattern match.

The given lists are matched each on the resulting set of complete records from the previous pattern-matching. The last filter applied will be accompanied by reduction of fields of final matching records as selected by "-o" option.

The main advantage of this approach is the simplicity of data structures and the utilization of common tools and data structures. Some performance gain is another advantage.

The drawback is, that in some cases the regexpr has to be choosen thoroughly.

Some Examples:

. (a single dot)
All items within the database.

inst
All items which contain any string "inst"

'^inst'
All items, which start with "inst", where the first field in a record is the hostname.

'*inst'
All items, which end with "inst".

'xen|qemu'
All items containing 'xen' or 'qemu'.

AND

The AND operator is the the same as a simple space-operator(" "), which causes the keyword to be applied as selective filter on the previous intermediate result. The result is matched based on the internal MACHINE format, which might lead to different results than the requested final output format only.

E:<#field0>:<#field1>
Compares two fields given by their canonical numbers.

The most important application might be the quer for a specific PM record, where the "netname" has to be matched by "PM", which is the "uname -n". Be aware, that only substrings and equal strings match, for local networks using DNS, the "netname" has an additional point "." at the end, thus order of numbers are significant for a match.

The "$<#field0>" is the canonical number as presented by TITLEIDX .

F:<#field0>:<content-match>
Queries for a specific FIELD with provided number to be compared by awk-function "match($<#field0>,<content-match>)". Be aware, that only substrings and equal strings match.

The "$<#field0>" is the canonical number as presented by TITLEIDX .

NOT
The NOT operator replaces the current composite state for the next argument only, operators are skipped. It should be recognized, that the NOT operator replaces only the current state, thus no chained evaluation of previous operators is applied. Anyhow, different operators, which are independant, such as NOT and AND, are superposed.

OR

The OR operator adds to the previous intermediate result a filtered subset of the last "AND-result". This sounds maybe a little strange, but simply said, a number of grouped OR operators just imply a parentheses/brace around all OR-ed elements. The overall operations is simple from-left-to-right.

The reason for omitting group-operators is just simplicity of implementation and grant of a resonable overall performance. When more operators are required, a full set of syntax might be implemented.

Exit Values:

- 0: OK

Result is valid.

- 1: NOK:
  Erroneous parameters.
- 2: NOK:
  Missing an environment element like files or databases.
- 7: NOK:
  Missing cacheDB directory.
- 8: NOK:
  Missing stat cache.
- 9: NOK:
  Missing groups cache.
- 10: NOK:
  Missing "macmap.fdb"
- 11: NOK:
  Unambiguity was requested by "-M unique", but query result is ambiguous.

18.13 ctys-vping

Usage:

```
ctys-vping [options] <group/machine-address>
```

Description:
ctys-vping uses ctys address resolution for resolving target addresses and validates accessibility by usage of "ping" only or when requested by combination of "ping" and "ssh" for assuring SSO access.

This particularly includes the usage of GROUPs including any options and context-options, where the resulting <exec-targets> are extracted and checked for their accesibility. In combination with the "-s" option this is a quite smart approach to check the SSH accesibility of the unmodified target list as supplied at the comman line or configured as GROUP.

SUBTASKs by keywords are not supported by this version.

Therefore the output is presented as a table, the input parameters are limited in comparison to native call of ping. Anyhow, the TCP/IP address is listed for additional usage of native ping, when extended analysis is required.

The entities will be resolved and accessed as selected by options. The two methods are "ping" which assures the basic TCP/IP-stack access, or the combined ssh-access, which performs a "ping" check first, following an remote call of "echo" by usage of ssh.

For each successful passed entity the basic ssh access and SSO configuration could be seen as validated and accessible when SSH is included.

Options:

-h
Print help.

-i
Displays the original raw input list of targets after expansion of GROUPs, the context options are displayed too.

-l <remote-user>
The remote user to be taken for ssh tests. Default is current user id.

-m
Output of MACHINE addresses in TERSE("-X") mode only.

-n
No checks are performed.

-r
Resolves not only "include" of GROUPs, but also SUBGROUPs.

-s
Activates additional SSH access test. When SSO is not configured properly, a password challange dialog occurs.

-t
Output of TCP/IP addresses in TERSE("-X") mode only.

-V
Version.

-X
Terse output format, effects "-V" when set left-of. When activated, the output is the resulting list as a simple space seperated string. In case of pre-checked mode successful checked entities are lsited only, remaining are suppressed.

Arguments:

<group/machine-address>
Any target to be checked by host, will be ordinarily ping-ed after ctys-name-resolution to TCP/IP-address.

Exit Values:

- 0: OK
  Result is valid.
- 1: NOK:
  Erroneous parameters.
- 2: NOK:
  Missing an environment element like files or databases.

18.14 ctys-wakeup

Usage:

```
ctys-wakeup [options] <MAC-address>
```

Description:
ctys-wakeup sends a so called MagicPacket(TM) to a given destination.

In case a remote segment is addressed by setting the "-t" option, a UDP message is sent to port "9".

The script consists of a few lines of bash code with a call to Netcat. So it could be adapted easily.

```
mac=$1;shift
ip=$1;shift

function buildWOLMagicPacket () {
 local _mac=$1;shift
 declare -a _pdu;

 function macAsc2Hex () {
   for i in ${@};do
     printf "\\\\x\$i"
   done
 }


 #frame
 _pdu=(ff ff ff ff ff ff);

 #MAC addr in "little endian"
 addr=(${mac//:/ })
 addrLitteEndian=(${addr[0]} ${addr[1]} \
                  ${addr[2]} ${addr[3]} \
```

```
                    ${addr[4]} ${addr[5]});


 #add 16 duplications
 for((i=0;i<16;i++));do
   for m in ${addrLitteEndian[@]};do
     size=${#_pdu[@]}
     _pdu[${size}]=$m
   done
 done

 macAsc2Hex ${_pdu[@]}
}

if [ -n "$ip" ];then
  printf "`buildWOLMagicPacket $mac`\\n"|nc -u -w 1 $ip 9
  else
    buildWOLMagicPacket $mac
fi
```

displays a short list of current available SMB server and workstations.

Options:

-d <level>
Sets debug.

-h
Print help.

-p <port>
The port for destination, which is by default "9" for a hard-coded UDP message. It is not really relevant, might be used only for adaptation to firewall rule.

-t <TCP/IP address>
When provided the packet is send to destination. The destination address could be a "directed-broadcast" address, which has to be supported by the router.

E.g. OpenBSD requires "net.init.ip.directed-broadcast=1", and the appropriate pf-rules.

For 1.2.3.4:

```
<wol>    {1.2.3.255}

pass  in  on \$ifX  from <wolExec> to <wol>
pass  out on \$ifY  from <wolExec> to <wol>
```

Alternatively a "rdr" - Redirection could be set.

When this option is not provided, a UDP packet will be sent to "255:255:255:255:9".

-V
Version.

-X
See ctys, terse for machine output.

Arguments:

<mac-address>
The MAC address of the targeted NIC, this could be evaluated by call of "ctys-macmap" and/or "ctys-vhost"

Exit Values:

- 0: OK
  Result is valid.
- 1: NOK:
  Erroneous parameters.
- 2: NOK:
  Missing an environment element like files or databases.

18.15 ctys-xen-network-bridge

Usage:

```
ctys-xen-network-bridge [options] [<action-arguments>]
```

Description:

This version is copied and adapted from the original script of Xen-3.0.3-25.0.4.el5 from CentOS-5.0 distribution for the UnifiedSessionsManager. This is preferred as an exceptional case, instead of a patch, because almost any second line is changed, even though for trace mainly.

The intention is not to take ownership or claim any rights.

The adaptation was neccessary, mainly due to debugging issues as an integral part for setting up WoL by PM, requiring a previous re-ordering and re-configuration of the IF targeted by WoL packets. The usage of the provided standard tool for Xen-ified machines was just the natural step. Anyhow, should be obsolete, when the so called MAGIC-PACKET(TM) could be set on a vif, other WoL-attributes, such as broadcast, seems to work already.

Copyright (C) by Owners of Xen
Copyright (C) by Owners of CentOS
Copyright (C) by RedHat Inc.

It is adapted in order of simplification of access grant for userland access. For ctys the "stop" case only is required when WoL has to be activated, thus this is the only verified call interface for now:

```
network-bridge stop bridge=<bridge-to-stop> netdev=<netdev>
```

This will be compatible for an immediate switch-over to the original script, when preferred. Some minor patches

for suppression of debugging flags might be required. For application and manual call the following has to be supported:

1. if appropriate: call only as root, than anthing might be ok.
2. if user land required:
   (a) set access bits for each PATH part of group for `/etc/xen/scripts/network-bridge`
   (b) assure ksu/sudo for at least: `ip, ifup, brctl`

Because this is just an error-workaround, for erroneous setting of "wol g" by ethtool, it may be removed later. For additional information refer to Xen documentation.

Options:
All options are "optional".

bridge=<virtual-bridge>
: The name of the virtual bridge to stop, for Xen this is frequently "xenbr0."

netdev=<original-network-device>
: The name of the interface to stop, for Xen this is frequently "eth0" or "bond0".

vifnum=<virtual-netdev-num>
: Normally not required, else frequently this is "0".

Arguments:

start
: Standard init argument.

status
: Standard init argument.

stop
: Standard init argument.

Exit Values:

- 0: OK
  Result is valid.

- 1: NOK:
  Erroneous parameters.
- 2: NOK:
  Missing an environment element like files or databases.

18.16 ctys-utilities

18.16.1 ctys-install

Usage:

```
ctys-install [<key>[=<value>]] ...
```

Description:
"ctys-install" is the base interface for installation and update. Therefore several call options are provided.

- initial install - common paths
  The "initial install" has no specific options, just the generic options for redefinition of install paths are available.
- update
  The options force, foceall, and forceclean are particularly available for protection or deletion of present user specific configuration files.
- common sources - symbolic links
  The "linkonly" option forces the installation of symbolic links pointing to the actual location of the called "ctys-install" directories and executables instead of performing a full scale install by copy.

  The usage of symbolic links for each executable has particularly advantages for "quick-installs" and test purposes when used in combination with a commmon directory structure on all machines by NFS. Thus multiple user accounts on multiple virtual and physical machines could be used for access to modified versions without requiring an update.

  The required configuration and template directories are still copied to the user's home for further modification. Several configuration files are initially present only as defaults within the resulting install directory and has to be copied by the user when required to be altered.

The following combinations of call options for install cover the common install procedures.

1. Individual Install
   Installs by copy of configurations and local lib-subdirectory containing the copy of installed version.

   - "ctys-install"
     The basic and initial call, when a complete local copy is required.
   - "ctys-install force"
     Suppresses the version check, which by default allows for updates with "higher" version numbers only. Thus the current version will be installed independetly of the previous.
   - "ctys-install forceall"
     When previous configuration or templates are present, by default the installation is canceled with a notification. When FORCEALL is set, the present directories are moved to a backup directoy by appending the current date to the directory name.

     The caution in handling this directories is due to the fact, that the configuration directories contain the individually edited configuratiob, GROUP and MACRO files, and the generated cacheDB files.
   - "ctys-install forceclean"
     The option FORCECLEAN forces the previous deletion of installed componnents, no backups are created.
2. Centralized Install by Local Symbolic Links
   Installs a copy of configurations and creates symbolic links to the install sources.
   - "ctys-install linkonly"
     The basic and initial call, where locall symbolic links and a minimal copy of configuration and template files is created. The actual runtime system is just called by symbolic links, where for each executable an individual link is created by default in "$HOME/bin".
   - "ctys-install linkonly force"
     See "ctys-install force"

- "ctys-install linkonly forceall"
  See "ctys-install forceall"
- "ctys-install linkonly forceclean"
  See "ctys-install forceclean"

3. Centralized Install by Search PATH
   Just install to any target, either by copy or by linkonly, and set the PATH as required.

Options

[libdir=<default=$HOME/lib>]

DEFAULT=$HOME/lib
The root path for the physical install target. The actual physical install is handled by a name including the current version in a similar manner as shared libraries naming convention.

[bindir=<default=$HOME/bin>]

The path to the starter directory, which is contained in the PATH variable.
Symbolic link: $bindir/ctys -> $libdir/ctys.<version>/bin/ctys
Bootstrap file: $bindir/bootstrap/bootstrap.<version>

[templatedir=<default=$HOME/ctys>]

Directory to templates and test-data matching current version. When an emtpy string is set, the install is suppressed, and the templates are contained within the libdir only.

[remove<default=unset>]

Removes previous configuration and templates.

[noconf<default=unset>]

Suppresses the install of initial configuration files. Could be somewhat dangerous, because some essential parameters are stored within the configuration and should match the executed version.

[force<default=unset>]

Checks "ctys -V -X" alphabetically/literally, normally only updates are allowed, but force installs in any case.

Current users configuration director $HOME/.ctys will be left unchanged.

[forceall<default=unset>]

Checks "ctys -V -X" alphebetically/literally, normally only updates are allowed, but forceall installs in any case.

Anything, else than users configuration directory $HOME/.ctys, will be removed and installed again. The users current configuration will be moved to $HOME/.ctys.bak.$DATETIME.

[forceclean<default=unset>]

Checks "ctys -V -X" alphebetically/literally, normally only updates are allowed, but force installs in any case.

Anything, including current users configuration directory $HOME/.ctys, will be removed and installed again.

[linkonly<default=unset>]

Suppress the local copy to $HOME/lib directory, just appropriate symbolic links are set to the given source directory determined by this call.

Anyhow, the configuration is copied and/or preserved as usual.

[version|–version|-V]

[-X]

[help|–help|-help|-h]

18.16.2 ctys-install1

Internal call for ctys-install.

18.16.3 getCurOS

Echoes the name of current OS. Will be used widely for internal execution variant selection during runtime.

18.16.4 getCurOSRelease

Echoes the release of current OS. Will be used widely for internal execution variant selection during runtime.

18.16.5 getCurDistribution

Echoes the name of current distro. Will be used widely for internal execution variant selection during runtime.

18.16.6 getCurRelease

Echoes the release of current distro-release. Will be used widely for internal execution variant selection during runtime.

18.16.7 pathlist

Shell interface for "pathSplit" library function, lists each element of common colon seperated search paths line by line. Could be used for trouble shooting support when PATH resolution fails.

18.16.8 ctys-getMasterPid

Internal call for evaluation of the master pid for a process. This works for detached processes and already running processes too. For additional explanation refer to Section 21.1.8 ‘procFindTopBottom’ on page 433 .

Part IV

Examples

Chapter 19

General Remarks

Even though this chapter should contain generic examples only, this is not possible due to the bare character of the basic framework when missing and plugin. So specific examples of a initial standard subset is provided. Even though in some examples the "-t" option is not set, this just causes the default type, which is e.g. VNC for CREATE and CANCEL action, will be used.

For additional information and examples refer to the previous sections for embedded examples within the plugins own documentation.

Chapter 20

ctys Setup

20.1 Installation

20.1.1 Basic Install

The current version of the UnifiedSessionsManager supports the install script ctys-install only.

The following steps has to be applied for installation:

1. Download
   The package is provided as a simple gzipped-tar file[150, UnifiedSessionsManager]. It has to be downloaded and unpacked. and unpacked to local filesystem.
2. ctys-install
   The install script in the binary directory of the unpacked package
   "ctys.<version>/bin/ ctys-install "
   has to be called. Several arguments could be applied, a list is displayed by "-h" option.

   An already present set of user specific configuration files is not removed by default, but could be forced to be replaced.
3. Check of required system-call permissions
   A number of required system-calls needs specific root call permissions, which has to be provided by one of the following means:
   - Execution as user "root"
     When using ctys as user root any required permission should be available.

- ksu
  "ksu" is the sudo like call of kerberos [125, MIT-Kerberos][126, Heimdal] , therefore ".k5users" and/or ".k5login" in the home directory of the permission-offering user(root) has to be stored.
- sudo
  Sudo[127, sudo] could be used aternatively or in combination with "ksu". Sudo provides a more fine-grained access administration, but is not(yet?) integrated into kerberos authentication. The file "/etc/sudoers" has to be edited appropriately by visudo.

  The potential pitfall when using SUDO is the requirement of a PTY by default, which could be configured within sudoers, but should be kept due to additional constraints by usage of OpenSSH. The options "-Z" and "-z" handle this issue.

Additionally the required tools such as bridge-utils has to be installed, as checked by the following calls.

The ctys-plugins tool supports a means to check (almost) any internal system call for it's presence and available call permissions. Some limit occurs on "inherent destructive" calls, which have no option to be used for check purposes only. Typicall for this are the shutdown-commands, which on some platforms could not be analysed for access permission without actually shutting down the machine.

The following calls could be used to validate the functionality of current installation:

- Check of Client Functions
  This checks the client set of required function calls.
  ctys-plugins -d 64,p -T all
- Check of Server Functions
  This checks the server set of required function calls
  ctys-plugins -d 64,p -T all -E

The previous calls utilize the internall common wrapper function

"checkedSetSUaccess"

by setting a specific debugging flag with the value "64=D_SYS". This activates the trace of system calls only. The options are described within the User-Manual, for addtional technical help on this library function call the online help for call interfaces could be used:

"ctys -H funchead=checkedSUaccess"

Any missing but required component as listed by the previous call should be installed and the required access granted by an appropriate entry for sudo/ksu.

4. VMW - Some specifics
   The installation of VMware is quite straight forward for supported Host operations systems by VMware.

   When installed on a non-listed OS, like CentOS, the required build of kernel modules during vmware configuration frequently causes some trouble. Simple warnings could be ignored in almost any case, but when errors occur the "vmware-any-any-update" patch from Petr Vandrovec[108, vmwareAny] should be applied. Some caution is required for usage of the appropriate version for matching the VMware version.
5. QEMU - Setup VDE
   When QEMU is going to be used, the VDE - Virtual Distributed Ethernet from VirtualSquare is required to be installed, therefore should be downloaded and installed by "make install". VDE is used for network setup of TAP devices to be used by QEMU with the utility "ctys-setupVDE" .
   The contained wrappers "vdeq/vdeqemu" are also required. For additional information on ctys wrappers for QEMU refer to Section 22.1.9 ‘Example Appliances’ on page 463 .
   A description howto proceed is given in Section 22.1.1 ‘Setup Networking for QEMU by VDE’ on page 446 .
6. XEN - Required packages and permissions
   Due to the CREATE and CANCEL actions, which start and stop a DomU, privileged access is required for the caller for permissions on functionality within the Dom0. Particularly to "xm create" and to "virsh". Therefore the virsh of libvirt is required additionally to Xen.

The current tested versions are Xen-3.0.x and Xen-3.1.x, but newer versions should work too. The main test platform is CentOS/RHEL, for this version only PVM tests are performed, but HVM might work too. The access permissions has to be set for sudo and/or ksu.

7. ctys configuration
   The next step should be the adaptation of the provided default configuration to local machine. Therefore the config file in the home directory of the user and/or the installed default files has to be edited. The provided configuration files contain the required description and additional examples.

8. Configure PMs
   The involved physical machines - PMs - should be configured by calling the tool ctys-genmconf in order to generate the PM configuration data. The result is stored by default within the directory "/etc/ctys.d". This step requires to be repeated when all hypervisors are installed and marked as operable by the check with ctys-plugins utility. This call updates the stored registration of capabilities available on the local machine, including the later required STACKCAP variable for verification of possible stacking capabilities, when a VM-Stack is going to be started. The update has to be performed before the creation of a cacheDB with the tool ctys-vdbgen .

9. Generate DHCP and/or MAC cache
   The access performance for plugins with stored configuration data - PMs and VMs - will be dramatically enhanced when generating a prefetched cache database. The first step required for the creation of the database is the creation of the mapping table generated by ctys-extractMAClst and/or ctys-extractARPlst , either from a valid "dhcpd.conf" of from simply "ping-ing" a list of hosts.

   The result consists of the data required for mapping of MAC addresses to IP/DNS addresses as provided by ctys-macmap . The format of stored data is documented and could be edited manually. It is an extended "/etc/ethers" format, the "/etc/ethers" database could

be generated by the previous tools.

10. Generate cache of configurations for VMs and PMs
    The access to stored static configuration is performed by a two step approach controlled by the option "-c" . The first attempt is to read from the generated cache, the second is to scan the filesystem on the target entity in accordance to the provided call options.

    The prefetch of the configuration data in a local file database enhances the call performance dramatically, a factor of at least 10, but almost in any case of 100 and much more is common.

    The build of the cache database is handeled by the tool ctys-vdbgen for collecting the data and by the tool ctys-vhost for preprocessing required correlations and intermediate preprocessings.

11. ctys
    Once this point is reached, ctys should be operational as required. The first tests should be executed by simple dynamic plugins such as CLI, X11, and VNC. Call examples are given in Section IV 'Examples' on page 413 .

    In case of errors the option "-d" provides a scalable degree of debugging information.

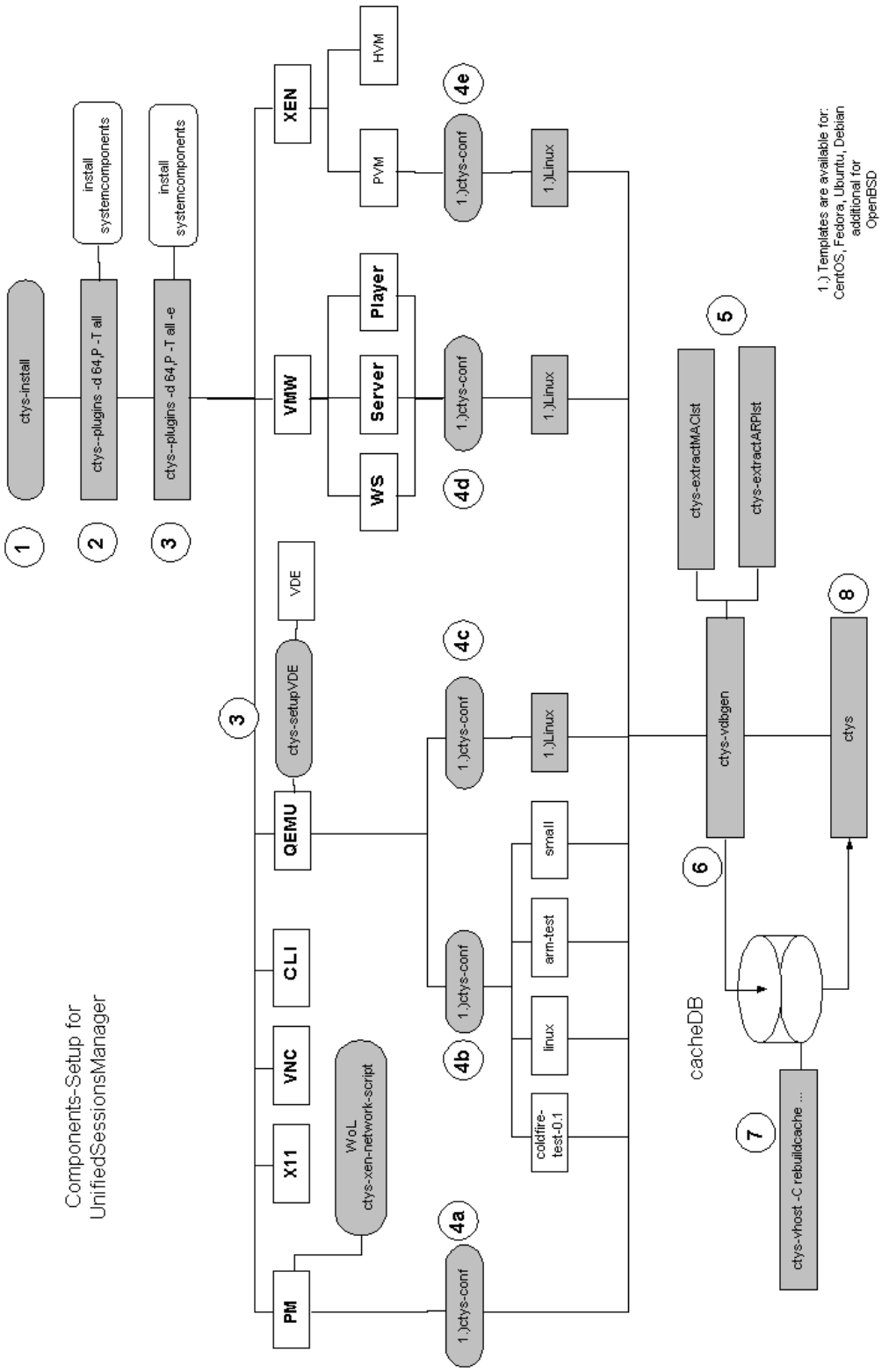

Figure 20.1: Install Steps

20.1.2 Security Environment

Some basic hints are given here only for the wide field of security and general access issues, additional support is available as commercial services only([?], [153], [152]).

General Remarks

Network Accounts
: When using UnifiedSessionsManager a network account with SSO is absolutly recommended. There are two exceptions, where the account should be a local account:
  - The root account sould be a local-only account, of course.
  - In case of required WoL for a machine running a virtual bridge, such as a Xen based machine, a local account is required. A "simple" shutdown may work without any difficulty. This is due to a currently required workaround particularly in case of Xen for setting of WoL on the physical NIC for offline operations of the NIC itself.

NFS
: Using NFS has some security risks, even though it is particularly a real benefit for development issues. Thus NFS versions <4 should not be used when crossing insecure network segments without additional provisions.

root Access
: Should be configured local only.

Router Configuration
: Router have to be configured for WoL, when the target NIC is outside the local segment. This is beyond the scope of this document.

Quick Setup for ssh

The only and one facility supported for communications between any type of entities is the usage of OpenSSH. Particularly the Handling of DISPLAY is deeply embedded into the UnifiedSessionsManager by usage of OpenSSH. Therefore the most benefit results from setting up SSH for SSO. The recommended setup is the usage of SSH in combination with Kerberos.

Quick Setup for ksu

The preferred authentication is the usage of Kerberos, which offers the access configuration facilities by ".k5login" and ".k5users". The usage is quite straight-forward, even though for some aspects not as flexible as "sudoers" is. Which is compensated by it's advance for networking purposes.

The only hard-wired restriction for usage of ksu applies, when network users are configured for CANCEL action on bridged NICs. When the WoL feature has to be applied, the NIC needs to be disconnected during the CANCEL procedure with continued access to restricted system resources. Thus this requires "sudoers" and a local user .

Quick Setup for sudo

Using sudo for authentication offers a perfect and straight forward setup for local users. For environments with NIS/NIS+ or any "distribution" utility a networked configuration update is available too. But anyhow, the initial access to a machine, and the "relay-user" is out of the scope of "sudoers".

The network access, particularly the SSO, is crucial for the applicability of ctys in a lager environment, thus sudo has to be used in companion with any network accounting facility only.

Quick Setup for LDAP

LADP is recommended for any distributed directory system. The user information should be handeled by LDAP, which has it's particular advantage when using a SMB based OS.

The setup in a heterogeneous environment is in details somewhat tricky, and requires some more description, though outside the scope of this document.

Quick Setup for autofs

Autofs is particularly the choice for a network login, when data has to be available within a secure segment only this could be simply based on NFS(version<4). The setup could particularly be based on LDAP in combination with Kerberos and SSH, with a lean centralized configuration. The description as provided by the project should suffice for the first steps.

Create a Local User

Even though almost anything could be configured to be accessed by networked users, the CANCEL action presets some additional requirements.

A local user is required for any CANCEL action on machines, where the network has to be disconnected as an intermediary step during the shutdown of the machine. Obviously, the process hangs, when some non-cached-modules has to be loaded via the disconnected connection, e.g. by NFS. This is required for now only for the Xen-3.0.x vesion, when setting WoL on a NIC, which is part of the virtual bridge, and may change for later versions. For details refer to Section 5b 'PM - Using Wake-On-Lan - WoL' on page 507 .

20.1.3 Setup Access Permissions

When the decision for the facilities to be used for authentication and authorization is made, and the configuration is finished, the specific required access permisssions for system calls by ctys could be established.

To check the actually present permissions for system calls the embedded debugging interface could be used. There are two basic approaches available:

- ctys-plugins
  This is the specific validation utility, which is a framework including some additional statistical display for the current states of plugins.

  ctys-plugins -d 64,P -T all -E

A second variant validates the local client functionality.

ctys-plugins -d 64,P -T all

- ctys

  ctys -d 64,P -T all -a list

  This lists all used system-calls with their actual checked execution states.

The check calls are actually the verified system calls encapsulated by a wrapper and performed with a more or less harmless option. The result is verified and weighted dependent of the actual call environment.

20.2 Configuration

20.2.1 Plugins

The following listed configuration file defines the actually used plugins. For example the VMW plugin is active for Linux OSs only, but neither when running on Solaris nor on OpenBSD.

For conceptual information refer to Section 10.1 'Basic Modular Design' on page 129 , particularly to the Section 10.2.2 'Operational States' on page 132 .

```
#When set, the bootstrap-loader ignores the given
# <plugin-type>.
#This should be in case of dependencies such as of XEN from
#VNC utilized carefully. But on machines with OpenBSD
#e.g. the plugins VMW and XEN could be set to ignore safely.
#
#export PM_IGNORE=1
#export CLI_IGNORE=1
#export X11_IGNORE=1
#export VNC_IGNORE=1
#export XEN_IGNORE=1
#export VMW_IGNORE=1
#export QEMU_IGNORE=1
```

```
#export OVZ_IGNORE=1
#
#
# This could be configured conditionally for each of the
# following variables, and any other valid shell variable.
# This is particularly helpful in case of NFS mounted home
# directories sharing the identical user-configuration for
# multiple machines within a cluster:
#
#   MYHOST   : actual host
#   MYOS     : actual OS
#   MYOSREL  : release of actual OS
#   MYDIST   : actual distribution
#   MYREL    : release of actual distribution
#

#
# Some examples for ignoring of XEN-plugin on specific
# sets of nodes:
#
#   ->host01 only
#     [ "$MYHOST" == "host01" ]&&export XEN_IGNORE=1;
#
#   ->Any node NOT on "clust0*"
#     [ "${MYHOST#clust0*}" == "${MYHOST}" ]\
#           &&export XEN_IGNORE=1;
#
#   ->Any node IS on "clust0*"
#     [ "${MYHOST#clust0*}" != "${MYHOST}" ]\
#           &&export XEN_IGNORE=1;
#
#   ->Any node in domain "exe1"
#     [ "${MYHOST##*.exe1}" == "${MYHOST}" ]\
#           &&export XEN_IGNORE=1;
#
#   ->Any node NOT running OpenBSD
#     [ "${MYOS}" != "OpenBSD" ]\
#           &&export XEN_IGNORE=1;
#
#   ->Any node IS running Linux
#     [ "${MYOS}" == "Linux" ]\
#           &&export XEN_IGNORE=1;
#
```

```
#   ->Any node IS running CentOS
#     [ "${MYDIST}" == "CentOS" ]\
#           &&export XEN_IGNORE=1;
#
#   ->Any node which IS NOT final execution target
#     REMARK: for now patched: [ -z "$CTRL_EXECLOCAL" ]\
#     [ -z "`echo $*|sed ....`" ]\
#           &&export XEN_IGNORE=1;
#
#   ->....
#
#
# Almost any combination and any additional constraint
# could be added by means of bash.
#
# When using internal state variables the user is
# responsible for any resulting side effect.
#


##################
#Configuration of supported contexts for standard
#plugins.
#

#PM supports currently Linux and OpenBSD
[ "${MYOS}" != "OpenBSD" -a "${MYOS}" != "Linux" ]\
&&export PM_IGNORE=1;

#XEN is supported on Linux only as server, else as
#client only.
#VNC check will be done by plugin
#Check for "-e", because CTRL_EXECLOCAL is not yet
#initialized.
[ "${MYOS}" != "Linux" -a -n "`echo " $* "|sed -n '/ -e /p'`" ]\
&&export XEN_IGNORE=1;

#VMW is supported on Linux only, else as client only.
#Native local client and VNC access for WS6 will be
#checked by plugin.

[ "${MYOS}" != "Linux" ]\
&&export VMW_IGNORE=1;
```

```
#############################################################
#                                                           #
#ATTENTION:                                                 #
#  The plugins CLI+X11+VNC could be called MANDATORY for    #
#  others, so their "IGNORE-ance" might force unforseen     #
#  side-effects, think twice!!!                             #
#                                                           #
#  Same is true for PM when you require WoL and             #
#  controlled PM shutdown, what might be obvious!           #
#                                                           #
#############################################################
```

Chapter 21

HOSTs - Sessions

21.1 CLI Examples

The CLI plugin itself is a pure command line interface, but due to the default activation of X11 forwarding any X11 command could be executed within a CLI session, thus any of the following examples could be used as a XTerm starter.

E.g. the interactive call of "xclock" will display correctly. This is particularly also true for the whole login-chain, when CLI is used for cascaded logins.

So, basically any firewall could be pierced in a secure manner by an SSH gateway in the DMZ. Which depends on the security facility of the gateway itself, of course.

HINT:

Spaces within options, including suboptions, have to be masked by the reserved character '%'.

This means just replace any SPACE with a % within any options suboptions.

21.1.1 Single Local Interactive Session

This opens a second shell as executed login, almost the same as an ordinary shell call.

```
ctys -t CLI -a create=l:test
```

The "localhost" is hard-coded to behave as sub-shell call too.

```
ctys -t CLI -a create=l:test localhost
```

REMARK:

Due to the implemented ambiguity-check for uniqueness of LABELs, only one localhost session is supported by the same label, when the label has not to be altered, the usage of "-A 1" disables ambiguity-check.

21.1.2 Single Remote Interactive Session

This opens a second shell as a remote executed login.

```
ctys -t CLI -a create=l:test lab00
```

21.1.3 Execute a Remote Command

This opens a remote shell and executes the provided command only before termination. The connection will be kept open during the whole session, thus is not executed in background mode.

```
ctys -t CLI -a create=l:test,cmd:uname%-a lab00
```

The same in background mode.

```
ctys -t CLI -a create=l:test,cmd:uname%-a \
     -b 1 lab00
```

21.1.4 Execute Multiple Remote Commands

The full scope of addressing of ctys is supported, thus the addressing of multiple targets, where each target could be a single host of a preconfigured hosts-group, is applicable. Intermixed addressing is supported too.

```
ctys -t CLI -a create=cmd:uname%-a lab00 lab01
```

The same with parallel background execution:

```
ctys -t CLI -a create=cmd:uname%-a -b 1 lab00 lab01
```

or

```
ctys -t CLI -a create=cmd:uname%-a \
     <host1> <group1> <host2> <group2>
```

The full scope of "include" files for group definitions and macros is applicable, thus e.g. tree-like reuse of groups could be applied.

Due to security reasons root-permission should be configured and handled properly, of course. Do not blame ctys, when failing.

21.1.5 Chained Logins by Relays

Even though there is upcoming a virtual circuit plugin, CLI could be used for digging multi-hop tunnels too.

It might be recognized that there is currently a chance (?) for users with appropriate permissions to intercept the communications, when on the intermediate hops the message flow has to be re-encrypted after decryption.

The vcircuit module under development will additionally encrypt the message-flow on an end-to-end basis.

The following call opens a session hop1 to lab01 via the intermediate relay lab00 by the session hop0.

```
ctys -t cli \
  -a create=l:hop0cmd:ctys%-t%cli%-a%create=l:hop1%lab00 \
  lab01
```

The following call opens a session hop1 to lab01 via the intermediate relay lab00 by the session hop0 and starts a Xterm on lab01.

```
ctys -t cli \
  -a \
   create=l:hop0cmd:ctys%-t%cli%-a%create=l:hop1,\
          cmd:xterm%lab00 \
  lab01
```

This approach is very similar to the equivalent usage of OpenSSH, and could be used in same manner to bypass routing as well as firewalls, when access and execution permissions on gateways are available.

21.1.6 Xterm with tcsh

This call starts an interactive XTerm session running tcsh inside.

```
ctys -t cli \
  -a create=l:tstcall,s:tcsh%-c,cmd:xterm%-e%tcsh \
  -b 1 \
  lab00
```

21.1.7 gnome-terminal

This call starts an interactive gnome-terminal session running tcsh inside.

```
ctys -t cli \
 -a create=l:tstcall,s:tcsh%-c,\
    cmd:gnome-xterminal%-e%tcsh \
 -b 1 \
     lab00
```

21.1.8 Call ctys functions

In a very similar syntax as the call of systems commands, any library function of ctys could be called within a CLI session. The whole set of addressing and execution functionality applies. But anyhow, most top-level interfaces require complex preparation of the call context, thus only helper methods or test-cases should be called.

The following call executes the internal function for evaluation of exeiting pairs of top-level task initiators and bottom-level "leafs" of worker threats for specific string-pattern, here plugin types.

procFindTopBottom

The internal libraryfunction "procFindTopBottom" lists for a given set of parameters the topmost and the bottom PIDs. The similar function "procGetMyMasterPid" utilizes this in order to evaluate the master-pid for the current process.

The generic commandline call interface is implemented as a utility process "ctys-getMasterPid" , which is a simple encapsulation of the basic call "procFindTopBottom" for usage within scripts. For additional help on "procFindTopBottom" call
"ctys -H funchead=procFindTopBottom" .

A typical application for this is given in the example "arm-test.ctys" for the QEMU plugin.

The utilized VDE/VirtualSquare version has the specific behaviour, that for a given switch for each call with "vdeq" creates an own port with it's own UNIX-Domain socket. This file socket is in current version not deleted when the port is closed by termination of the clien VM and could lead in some cases to difficulties when assigned again, which particularly could happen during excessive tests of CREATE and CANCEL actions. The socket name itself is constructed by usage of the PID of "vdeq" process used as the wrapper for QEMU VM

call. This wrapper-mechanism, as the whole process structure, is pre-required for usage of the "vde_switch" component in combination with qemu. I honestly do not want to miss this!

Thus as a temporary workaround the PID of the vdeq-process is required to be evaluated. Therefore the internal function "procFindTopBottom" is utilized.

The second application is the evaluation of the socket for the port used as local accessport for Monitor functions. Therefore the master-pid is used to generate the pathname.

```
CALL:
 ctys \
  -t cli \
  -a \
   create=l:tst1,\
    s:procFindTopBottom%ctys%%vdeq%ctys%vdeq%w%%25832 \
  lab00

or

 ctys \
  -t cli \
  -a \
   create=l:tst1,\
    s:procFindTopBottom%ctys%%vdeq%ctys%vdeq%w%%25831 \
  lab00

or

 ctys \
  -t cli \
  -a \
   create=l:tst1,\
    s:procFindTopBottom%ctys%%vdeq%ctys%vdeq%w%%25704 \
  lab00

RESULT:
  ctys:25704:vdeq:25832
```

The following has to be considered:

- "%" and "%%"
  The percent signs are used within the UnifiedSessionsManager as a padding character to be replaced by a space on the final target. Space within arguments and options are not supported, these will be interpreted as seperators. The double usage of percent "%%" masks the character itself and is replaces on the final target by one percent character "%". So the following parameters result as final call parameters for the function:

  procFindTopBottom ctys%vdeq ctys vdeq w%25832

  Thus the calls "ctys" or "vdeq" are scanned for toplevel match of "ctys" and bottomlevel match "vdeq".
- 25832
  This is the PID of the vdeq process call chain, where the instance is contained and the related "master-PID" has to be evaluated. The "master-PID" is the pid of the topmost process with matching "ctys" as top-label. Thus in case of mandatory usage of SSH for ctys, it is the first non-ssh-process forked and executed by the sshd.

  The concrete value, as the presented PIDs in the result depends on the actual operated context of environment, of course. The same for the following examples.

  In case of QEMU atleast a three-level structure is (for now) foreseen above "vdeq", thus in the test example the following structure was present.

PID	PPID	Process Name	Example-Scope
25704	1	arm-test.ctys	X
25831	25704	arm-test.ctys	X
25832	25831	vdeq	X
25834	25832	qemu-system-arm	-

  The application here is the evaluation of the PID of the "vdeq" wrapper and it's master-pid for the calculation of the name for the UNIX-Domain socket of the port on "vde_switch" as described before and the dynamic part of the monitoring socket. The bottom PID is the dynamic part of "SPORT" attribute of the "LIST" action output.

The following call shows both, the PID of the QEMU session, which is defined as the "vdeq" pid, and the SPORT of the QEMU session, which is defined as the master-pid.

```
ctys macro:listconnpid lab00
```

For additional information on MACROs refer to Section 6.2 'CLI-MACROS' on page 73 .

```
label   |stype    |c|DIS|cport|sport|pid   |PM         |TCP
--------+--------+-+---+-----+-----+-----+----------+------------
LAB00   |VNC      |C|1  |     |     |18933|lab00.soho|
LAB00   |VNC      |S|1  |5901 |     |5642 |lab00.soho|
arm-test|QEMU-arm|S|17 |     |25704|25832|lab00.soho|
Domain-0|XEN      |S|   |     |     |     |lab00.soho|
lab00   |PM       |S|   |     |     |1    |lab00.soho|192.168.1.71
```

The next example is an arbitrary test call for the CLI plugin.

```
CALL:
  ctys -t cli \
   -a create=l:tstcall,s:procFindTopBottom%ctys%CLI%CLI \
   lab00

RESULT:
  CLI:3784:CLI:3784 CLI:7430:CLI:9089 CLI:20769:CLI:20771 \
  CLI:31206:CLI:31206
```

The "pstree" command visualizes the structure.

The same for collecting any X11 session which is running as a XEN client.

```
CALL:
  ctys -t cli \
   -a create=l:tstcall,s:procFindTopBottom%ctys%XEN%X11 \
   lab00

RESULT:
  XEN:26471:X11:28121 XEN:26471:XEN:28121
```

21.2 X11 Examples

21.2.1 Xterm with Interactive bash

This opens a Xterm window with an interactive bash. The "SH" suboption is here mandatory, because the Xterm emulation requires a single-hyphen for it's options, default is "DH".

The default behaviour concerning the terminal emulation is to scan for a gnome-terminal first and prefer it if found, else an xterm emulation will be started by default.

```
ctys -t x11 \
 -a create=l:tstx11 \
 lab00
```

The same by call CONSOLE:XTERM alias.

```
ctys -t x11 \
 -a create=l:tstx11,console:xterm \
 lab00
```

The same by call args.

```
ctys -t x11 \
 -a create=l:tstx11,cmd:xterm,sh \
 lab00
```

The same by call args with explicit shell.

```
ctys -t x11 \
 -a create=l:tstx11,cmd:xterm,c:bash%-i,sh \
 lab00
```

21.2.2 Gnome-Terminal with Interactive bash

This opens a Gnome-Terminal window with an interactive bash. The "DH" suboption is here mandatory but the default. This is required because the gnome-terminal emulation requires a double-hyphen for it's options.

The the default behaviour is to open a gnome-terminal when detected.

```
ctys -t x11 \
 -a create=l:tstx11 \
 lab00
```

The same by call CONSOLE:GTERM alias.

```
ctys -t x11 \
 -a create=l:tstx11,console:gterm \
 lab00
```

The same as the default behaviour.

```
ctys -t x11 \
 -a create=l:tstx11,dh \
 lab00
```

The same as the default behaviour.

```
ctys -t x11 \
 -a create=l:tstx11,cmd:gnome-terminal,c:bash%-i,dh\
 lab00
```

21.2.3 Emacs Session in shell-mode

Starts an remote Emacs in

```
ctys -t x11 \
 -a create=l:tstx11,cmd:emacs%-f%shell,s:bash%-i \
 lab00
```

The same by call CONSOLE:EMACS alias.

```
ctys -t x11 \
  -a create=l:tstx11,console:emacs lab00
```

21.2.4 Single XClock

This starts an Xclock with an altered options keyword for windows title, additionaly the "SH" key is required for using single-hyphens.

```
ctys -t x11 \
 -a create=l:tstx11,cmd:xclock,titlekey:name,sh \
 lab00
```

Similar, but not the same. This starts an Xclock with NOTITLE, which suppresses the setting of the title for a window by the LABEL string. Thus this window is no longer recognized by LIST, or just with limits.

```
ctys -t x11 \
 -a create=l:tstx11,cmd:xclock,notitle \
 lab00
```

21.3 VNC Examples

The automated signon when connecting a vncviewer to a vncserver requires a password as supported by vncpassd. In order to avoid any user interaction for password requests the password is stored into the passwd file in "$HOME/.vnc" and is set to a default "install". This has to be changed once installed.

Do not forget to set the appropriate attributes in "/etc/ssh/sshd", particularly "X11Forwarding yes" has to be set, which is "no" by default. For using root during installation and basic configuration eventually "PermitRootLogin yes" should be set.

21.3.1 Single Local Desktop Session

This opens a local session, where the VNCserver as well as the VNCviewer are executed locally.

```
ctys -t VNC -a create=l:tst1
```

The "localhost" is hard-coded to behave as a sub-shell call too, thus the following call is internally handeled identical to the previous

```
ctys -t VNC \
 -a createl=l:tst1 \
 $USER@localhost
```

The distribution comes with the default setting of "-t VNC" option. Thus the following call is identical to the previous.

```
ctys -a create=l:tst1
```

21.3.2 Single Remote Desktop Session

This call opens a remote desktop with DISPLAYFORWARDING, which is a coallocated VNCserver with a VNCviewer on the <execution-target>.

```
ctys -a create=l:tst1 \
 -L DISPLAYFORWARDINGF \
 <host>
```

The same could be written as:

```
ctys -a create=l:tst1 -L DF lab00
```

The Client-Location "-L DISPLAYFORWARDING" is default for the original distribution, thus could be written as:

```
ctys -a create=l:tst1 lab00
```

21.3.3 Bulk Desktop Sessions

This call opens 3 desktops on the remote host. The internal limit is set by default to 20.

```
ctys -a create=bulk:3,l:tst lab00
```

The following call cancels all session by addressing their labels. The complete label is required here, which is an extended label by the incremental bulk-counter.

```
ctys -a cancel=l:tst000,l:tst001,l:tst002 app2
```

The same function with usage of IDs.

```
ctys -a cancel=i:2,i:3,i:4 app2
```

Current version supports as an implicit bulk addressing the keyword "ALL" only, which kills literally all VNC session where the appropriate permissions are available.

```
ctys -a cancel=all app2
```

It should be recognized, that the CANCEL action is just a call to "vncserver -kill <display>" command, when this does not succeed, a "kill" will be placed. The clients are killed by UNIX-calls when required.

So the user is responsible for shutting down applications running within the CANCEL-ed sessions.

21.3.4 Remote Desktop with Local Client

In case of a "Remote Desktop with Local Client" the server is started on the given <execution-target>, whereas the client is locally started on the caller's machine. This structure is called CONNECTIONFORWARDING and requires beneath the client and server processes a third, the connecting encrypted tunnel. The tunnel is established by means of OpenSSH and used as the local peer for the Client. This whole procedure of starting the processes and the establishment of the tunnel is controlled and preformed by ctys.

The scenario differs in all steps except the start of the server process from the previously described DISPLAYFORWARDING structure. In case of CONNECTIONFORWARDING the whole process is set up in three steps.

1. start of server process
2. establishment of the encrypted tunnel
3. start and connect the client process to the tunnel

The tunnel is established in the so called one-shot mode, by where connection is opened for an inital time period and closes automatically when the life-time threshhold reached, or afterwards, when the server disconnects. The period of the initial lifetime is defined by the variable "SSH_ONESHOT_TIMEOUT", which is by default set to 20seconds.

The following call starts a remote server with a local client.

```
ctys -a create=l:tst -L CF lab00
```

The instances could be listed by the LIST action in several variants.

The basic call with default selection executed on the caller workstation is:

```
ctys -a list ws2
```

The standard assignement to LIST call is "tab_tcp,both", which displays:

```
TCP-container|TCP-guest|label |sesstype|c|user|group
-------------+---------+------+--------+-+----+---------
ws2.soho     |-        |tst000|VNC     |C|acue|ldapusers
ws2.soho     |-        |tst001|VNC     |C|acue|ldapusers
ws2.soho     |ws2.soho.|ws2   |PM      |S|-   |-
ws2.soho     |-        |tst000|SSH(VNC)|T|acue|ldapusers
ws2.soho     |-        |tst001|SSH(VNC)|T|acue|ldapusers
ws2.soho     |-        |tst000|VNC     |C|acue|ldapusers
ws2.soho     |-        |tst001|VNC     |C|acue|ldapusers
```

Here the two tunnels could be identified as "sesstype=SSH(VNC)", and "c=T". This indicates, that the tunnels are created for the subsystem VNC with the session label "tst000" and "tst001".

The following call displays the same table, but with IDs instead of LABELs.

```
ctys -a list=tab_tcp,id ws2
```

Which results to the display:

```
TCP-cont|TCP-guest|id        |sesstype|c|user|group
--------+---------+----------+--------+-+----+---------
ws2.soho|-        |50        |VNC     |C|acue|ldapusers
ws2.soho|-        |51        |VNC     |C|acue|ldapusers
ws2.soho|-        |../pm.conf|PM      |S|-   |-
ws2.soho|-        |5950-5903 |SSH(VNC)|T|acue|ldapusers
ws2.soho|-        |5951-5904 |SSH(VNC)|T|acue|ldapusers
ws2.soho|-        |50        |VNC     |C|acue|ldapusers
ws2.soho|-        |51        |VNC     |C|acue|ldapusers
```

Indicating by the default ID of tunnels, that these are tunnels forwarding the ports "5950" to "5903" and "5951" to "5904".

The display could be changed as required by usage of specific free-customized tables, e.g. displaying LABEL and ID columns once.

The call with the whole set of involved machines as one call results to:

```
ctys -a list=tab_tcp,id ws2 lab00 lab01
```

```
ctys -a list=tab_tcp,id ws2 lab00 lab01
```

```
TCP-contai|TCP-guest|id         |sesstype|c|user|group
----------+---------+-----------+--------+-+----+---------
ws2.soho  |-        |50         |VNC     |C|acue|ldapusers
ws2.soho  |-        |51         |VNC     |C|acue|ldapusers
ws2.soho  |-        |d/pm.conf  |PM      |S|-   |-
ws2.soho  |-        |5950-5903  |SSH(VNC)|T|acue|ldapusers
ws2.soho  |-        |5951-5904  |SSH(VNC)|T|acue|ldapusers
lab00.soho|-        |3784       |CLI     |C|acue|ldapusers
lab00.soho|-        |31206      |CLI     |C|acue|ldapusers
lab00.soho|-        |1          |VNC     |S|root|root
lab00.soho|-        |2          |VNC     |S|acue|ldapusers
lab00.soho|-        |           |XEN     |S|-   |-
lab00.soho|-        |e/xen/tst1 |XEN     |S|-   |-
lab00.soho|-        |d/pm.conf  |PM      |S|-   |-
lab01.soho|-        |           |XEN     |S|-   |-
lab01.soho|-        |d/pm.conf  |PM      |S|-   |-
```

21.3.5 Shared-Mode vs. Non-Shared-Mode

21.3.6 Reconnect Suboption

Chapter 22

VMs - Sessions

22.1 QEMU Examples

22.1.1 Installation of QEMU

Install rpm

The installation of the RPM package is quite forward. At least for Fedora the paths are unchanged from the original "configure" output.

Install from Sources - Build and Install QEMU

The QEMU installation itself is quite straight forward, the only point to be careful about is the requirement of a gcc3x compiler, gcc4 is not compatible. This is required for the configuration and compilation only, thus the installation could be performed on gcc4x.

The kqemu component was not installed for now, because some VM stack tests with nested execution are required, and side-effects should have been excluded. Use the "-no-kqemu" flag for the qemu-call for version >0.9.1.

Previous of installation the build from the sources was executed. Therefore a 3x version of gcc is used as required. Due to the lack of a native install a VMware installation of SuSE-9.3 was used.

BUILD
On SuSE-9.3 running in VMware with gcc3.x:

1. Install gcc3x and SDL, un-tar "qemu-0.9.1.tar.gz"
2. call "configure"
3. call "make"
4. copy to CentOS, the paths has to be the same, due to "configure" resolution for "make install", so choose a common or network directory, such as "/tmp" or "/mntn" for build.

INSTALL
On target system, here CentOS-5.0 with gcc 4.x:

1. call "make install"

Verify
On the execution-target machine, this is the machine running the actual QEMU-VM, call

"ctys-plugins -T all -E"

and on the client machine

"ctys-plugins -T all"

When the last error message complains the absence of QEMUSOCK and QEMUMGMT, than anything seems to be perfect, just missing the final call for network setup by "ctys-setupVDE" .

Setup Networking for QEMU by VDE

Overview
Now, once the "make install" is performed, the base package of QEMU is installed. As mentioned before, here the kqemu package is not installed, because stacked VMs and heterogeneous CPU environments are going to be tested.

The next step now is to create a runtime environment and install the BIOS binaries and test examples for later validation of QEMU.

The expected runtime structure is as described before:

```
mkdir -p $HOME/qemu/{ctys,bc-bios/keymap,tst}

cp -a $HOME/.ctys/templates/qemu/tst/* \
   $HOME/qemu/tst
```

Check the x-bit of all contained configuration scripts, which are bash-scripts actually to be called as executable.

Now copy and unpack the appropriate test images from the qemu distribution into the tst-subdirectories.

- arm-test-0.2.tar.gz
- coldfire-test-0.1.tar.bz2
- linux-0.2.img.bz2
- mipsel-test-0.2.tar.gz
- mips-test-0.2.tar.gz
- small.ffs.bz2
- sparc-test-0.2.tar.gz

These images are ready-to-use VMs. The only configuration required later is the setting of appropriate IP address, except for the coldfire image, which is based on DHCP. This is only true if you are using DHCP and have set the appropriate MAC addresses of course.

The usage of bridged network with communications via the NIC of the host requires some additional effort. Particularly the creation of the required TAN-device with the frequently mentioned "tunctl" utility from the "UserModeMLInux" was somewhat difficult on CentOS-5. This results particularly from the missing tool "tunctl".

The package "vde" including a (not-found-documentation-for)" utility "vde_tunctl" was the rescue-belt.

So proceed as described in the common examples Section 30.1 ‘TAP/TUN by VDE’ on page 579 for your matching environment in order to create the required TAP device and the

appropriate "vde_switch".

Now edit the configuration files within the tst subdirectories and adapt the MAC addresses, IP-Addresses, and UUIDs as required. They might be usable as given, if no collisions with existing addresses occur.

At this point anything might be prepared for successful operations.

Now test the installation:

1. raw-call without network for linux-subdirectory.

   ```
   qemu linux-0.2.img
   ```

2. raw-call with network

   ```
   vdeqemu -vnc :17 -k de -m 512
   -hda linux-0.2.img
   -net nic,macaddr=<mac>
   -net vde,sock=/var/tmp/vde_switch0.<user>
   ```

   Within GuestOS:

   ```
   ifconfig eth0 <your-ip>
   ping -c 5 <known-host>
   ```

   Should show what you expect. OK, if so, than you've got it!

3. encapsulation ctys-script-call
   "./linux.ctys SDL"
   Might work now.
4. ctys call

VDE
After the "some hours" of trial-and-error the VDE project (see [131, VirtualSquare] and [130, VDE]) was found and installed shortly before giving it up.

After some straight-forward steps now - take due to own mistakes for about 15minutes - anything worked perfectly.

Several combinations of VDE are installed and configured in order to create a TAP device. The various cases are listed due to their generic character within the chapter containing the common examples.

The following cases are shown in the examples section:

- VDE within Xen Dom0
- VDE within Xen DomU
- VDE within VMware
- VDE within Native Unix

QEMU
For Qemu several excellent sites with install descriptions exist in the net. Here are just some shortcuts, which seem to be the most important items.

PXE-Boot

The following applies to a configured PXE environment based on PXELinux.

```
vdeqemu -vnc :17 -k de -m 512
        -hda linux-0.2.img
        -net nic,macaddr=<mac>
        -net vde,sock={QEMUSOCK}
        -boot n
        -option-rom ${QEMUBIOS}/pxe-ne2k_pci.bin
        \&
```

ISO-Image-Boot

The following applies to a boot CDROM image for installation.

```
vdeqemu -vnc :17 -k de -m 512
        -hda linux-0.2.img
        -net nic,macaddr=<mac>
        -net vde,sock={QEMUSOCK}
        -boot d
        -cdrom ${QEMUBASE}/iso/install.iso
        \&
```

shared memory
: Should be set in "/etc/fstab" as required. This value might be raised, when nested stacks are used, thus the bottom engine requires the sum of the resources of the stack. The entry might look like:

  ```
  none /dev/shm tmpfs defaults,size=512M 0 0
  ```

  For a call like:

  ```
  vdeqemu -m 512 ...
  ```

  The changes could be activated with

  ```
  mount -o remount /dev/shm
  ```

Create image
: To create an ISO image a call like the following could be applied

  ```
  qemu-img create -f qcow myImage.qcow 4G
  ```

  which could be used as

  ```
  qemu -cdrom installMedia.iso \
    -boot d myImage.qcow
  ```

Setup Serial Console

The setup of a serial console for QEMU is required for various CONSOLE types. Any CONSOLE providing an ASCII-Interface, except the syncronous un-detachable CLI console, requires serial access to the GuestOS. This is a little complicated to setup for the first time, but once performed successful,

it becomes an easy task for frequent use.

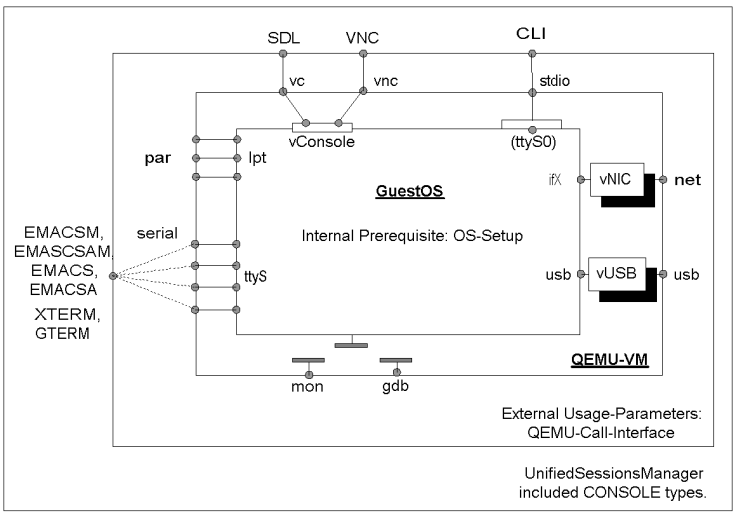

Figure 22.1: QEMU Interconnection-Interfaces

The first thing to consider is the two step setup, which comprises the initial installation with a standard interface either by usage of SDL or by usage of the VNC console. The second step - after finishing the first successesfully - is to setup the required serial device within the GuestOS. This requires a native login as root. Detailed information is for example available at "Linux Serial Console HOWTO"[137, VANEMERY], "Serial-HOWTO"[138, GHAWKINS], and "Text-Terminal-HOWTO"[139, DSLAWYER].

The following steps are to be applied.

1. Install GuestOS by usage of SDL/VNC as console.
2. Login into the GuestOS.
3. Adapt "/boot/grub/menu.lst"

   The header section:

   ```
   serial --unit=0 --speed=9600 --word=8 --parity=no --stop=1
   terminal --dumb serial console
   splashimage=(hd0,0)/grub/splash.xpm.gz
   ```

```
#
default=<#yourKernelWithConsole>
#
```

Your target kernel for boot <yourKernelWithConsole>:

```
kernel ... console=tty0 console=ttyS0,9600n8
```

4. Adapt "/etc/inittab", here for CentOS-5.0

   ```
   S0:12345:respawn:/sbin/agetty ttyS0 9600 Linux
   ```

5. Adapt ctys-qemu-wrapper
   This requires adapted items, which depend on the choosen CONSOLE type.
   - A CLI, which is a synchronous console, requires:

     `-nographic`

     This switches off the graphic and defaults it's IO to the caller's CLI.
   - A "by-Window-Encapsulated-CLI", which is a non-"modal" console, called as a serial console by a UNIX-Domain socket within a X11-Window/Client, requires:

     ...

     This switches off the graphic and defaults it's IO to the caller's CLI.
   - Reboot.

22.1.2 Installation of Guests

PXE-Boot

The PXE based installation is possibly not the fastest, but it offers a common neatless solution for unified installation processes. Even though an image could be just copied and modified as required, some custom install procedures might be appreciated, when the install could be performed in batch-mode. One example is the usage of kickstart files for CentOS/RHEL. In case of PXE these files are almost the same for any install base, this spans from physical to virtual machines.

Install Procedures

Debian-4.0_r3 by PXE - i386
The template "debian-i386" contains particularly the BOOTMODES PXE and INSTALL.

Debian-4.0_r3 for ARM
The installation of debian for ARM9 is quite good described for example by the online articles of Aurelien Jarno[93] and Brian Dushaw[95]. The following description is based on [93, AurelienJarno], which was adapted to the usage of VDE.
For analysis and detailed understanding of the required installprocedures of debian on a non-standard platform for an emulated CPU, this is explained first as a manual procedure based on [93].

(a) download pre-requisites
Even though the kernel and the initrd could be cross-compiled from the scratch, the prepared variants are used here, these could be downloaded from [94, AurelienJarnoPrecomp], for additional information refer to [93].

This description is based on the "etch" distribution "debian-4.0_r3". The ISO images are used on a local iso server by http protocol, where the authentication has to be suppressed due to missing "Release.gpg" file. The following files are required(versions may vary) in addition to the ISO images.

- vmlinuz-2.6.16-6-versatile
- initrd.img-2.6.16-6-versatile
- initrd.gz(for install only)

(b) create image for vhdd

```
qemu-img create -f qcow hda.img 10G
```

(c) start installation
The following call is based on the previous setup of VDE by execution of "ctys-setupVDE" .

```
vdeq qemu-system-arm -M versatilepb \
```

```
-kernel vmlinuz-2.6.16-6-versatile \
-initrd initrd.gz \
-hda hda.img \
-k de \
-net nic,macaddr=00:50:56:13:11:42 \
-net nic,sock=/var/tmp/vde_switch0.tstusr \
-append "root=/dev/ram \
-- debian-installer/allow_unauthorized=true"
```

(d) installation steps
The intstallation itself should be proceeded as described in [93, AurelienJarno] the references. Just some minor issues remain:

- keyboard
  During install some keys are not mapped correctly, for the german keyboard within the test environment this is "/", which is actually printed by "Shift-6".
- expected-install errors
  Due to missing native kernel, the external kernel and initrd are used by call parameters, for install and runtime. Therefore any error message related to misisng kernel on installed system could be ignored safely.
- finish install
  The final step has to be "finished" but not "cancelled", and when the VM automatically reboots for first time, it has to be "stopped". Preferably by usage of the QEMU-monitor. When cancelled the installed image my be corrupt and hangs during boot at fsck.

(e) boot runtime-system
Once the system is installed, the install-initrd has to be replaced by the runtime specific image and the root device needs to be changed to the runtime image.

The example values has to be adapted, of course.

```
vdeq qemu-system-arm -M versatilepb \
  -kernel vmlinuz-2.6.16-6-versatile \
  -initrd initrd.img2.6.16-6-versatile \
```

```
-hda hda.img \
-name debian-arm-demo \
-k de \
-net nic,macaddr=00:50:56:13:11:42 \
-net nic,sock=/var/tmp/vde_switch0.tstusr \
-append "root=/dev/sda1"
```

OpenBSD-4.3+SerialConsole - by PXE
The installation is straight forward(refer to template for tst127, BOOTMODE=PXE). When for later access a serial console is required following additional steps has to be proceeded. A serial console is a prerequisite for EMACS, XTERM, and GTERM.

(a) Boot and login into GuestOS if the root-filesystem cannot be mounted offline.

(b) Create a file "/etc/boot.conf" with content such as:

```
set tty com0
stty com0 115200
set timeout 15
boot
```

(c) Edit the file "/etc/tty" and change the lines:

```
tty00   "/usr/libexec/getty std.115200" vt220 on secure
```

Installed Systems

OS	name	Inst-VM
CentOS-5.0	tst000	0.9.1
Debian-4.0r3-arm	tst102	0.9.1
Debian-4.0r3-i386	tst108	0.9.1
Fedora 8	tst105	0.9.1, 0.9.0
OpenBSD-4.0	tst124	0.9.1
OpenBSD-4.3	tst127	0.9.1
SuSe-10.3	tst113	0.9.1
uClinux-arm9	(QEMU)	0.9.1
uClinux-coldfire	(QEMU)	0.9.1

Table 22.1: Overview of Installed-QEMU-VMs

In addition various test packages with miscellaneous CPU emulations of QEMU are available. Example templates for

integration scripts are provided for ARM, Coldfire, MIPS, and PPC.

22.1.3 CREATE a session

The first tests and examples of the QEMU plugin are based on the "arm-tst" VM contained in the examples of QEMU. This is a ready to use VM, but the TCP/IP address is hardcoded to "10.0.0.2" thus might be required to be configured. The coldfire test VM contained in the QEMU examples supports DHCP, thus is ready to use within the network.

Anyhow, for the first tests the actual usage of the network is not yet required. All following examples, if not stated else, rely on the provided configuration file "arm-test.ctys" and the QEMU VM "arm-test". Theses have to be installed as described within the examples chapter Section 22.1.1 'Installation of QEMU' on page 445 .
The first call now creates a session and starts the VM with VNC as a console which will be attached automatically.

```
ctys \
 -t qemu \
 -a
 create=p:$HOME/qemu/tst/arm-test/arm-test.ctys,\
   reuse,console:vnc
 lab00
```

When the "vde_switch" is not configured yet the following error message occurs:

```
Missing management socket for "vde_switch"
   QEMUMGMT=/var/tmp/vde_mgmt0.acue
Call: "ctys-setupVDE" on "lab00.soho"
```

The solution is simply to proceed as requested and just create the UNIX Domain sockets by the following call:

```
ctys-steupVDE -u $USER create
```

This call requires root permissions.

That's it.

The support of the types of CONSOLEs depends on the actually implemented call within the "arm-tst.ctys" script, which is a shell script with a defined interface. The currently supported CONSOLE types by arm-test are: "CLI, SDL, VNC". The CLI and SDL types are supported as DISPLAYFORWARDING in synchronous mode only for this version.

The following call creates a SDL CONSOLE.

```
ctys \
 -t qemu \
 -a
 create=p:$HOME/qemu/tst/arm-test/arm-test.ctys,\
   reuse,console:sdl
 lab00
```

As might be expected, the following call creates a CLI CONSOLE.

```
ctys \
 -t qemu \
 -a
 create=p:$HOME/qemu/tst/arm-test/arm-test.ctys,\
   reuse,console:cli
 lab00
```

The monitor as configured within "arm-test.ctys" could be attache by usage of "netcat"

```
nc -U ${MYQEMUMONSOCK}
```

which could be generated by function "netGetUNIXDomainSocket" and is derived from "QEMUMONSOCK" as raw-pattern, for additional information refer to the "arm-text.ctys" inline comments.

The QEMU monitor now could be entered by typing "Ctrl-a c RET", the console is recovered by typing the same sequence again. For additional information refer to the QEMU User-Manual.

A second terminal emulation to be used is the "unixterm" command of VDE.

Alternatively EMACS could be used as a terminal emulation for CONSOLE access, either in "shell-mode" or in "ansi-term" mode. This work s the same way as an ordinary xterm session, where within the the "display-window" a cli is started connecting to a local UNIX domain socket. The socket has to be configured as a serial device within the GuestOS. For EMACS two variants additional variants exist for both modes, where the frame is divided into two windows, which connect the <execution-target> and the <machine-address> representing the GuestOS.

In the following example in the upper window a login-prompt of the GuestOS is displayed, whereas in the bottom window the "top" comman is shown for the hosting machine.

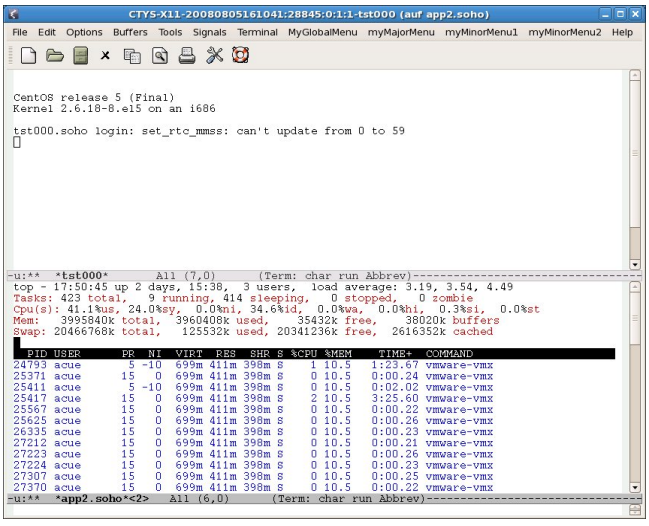

Figure 22.2: The "CONSOLE:EMACSAM" for a QEMU Session

The console with pure CLI access could be combined with

an VNC console allowing additional graphical access. This is particularly forseen, and woll be offered soon, ad a debugging facility for GDB access to QEMU and to applications within the GuestOS.

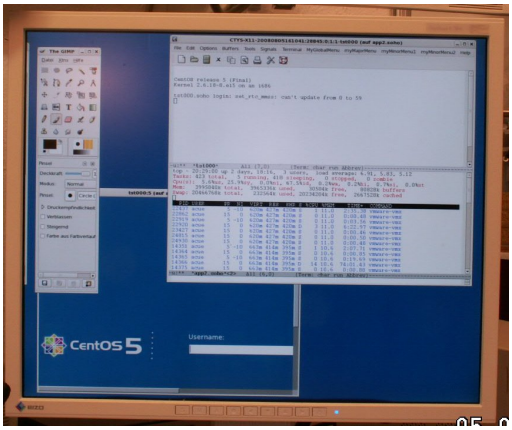

Figure 22.3: The "CONSOLE:EMACSAM" and a "CONSOLE:VNC" for a QEMU Session

22.1.4 CANCEL a session

The following call CANCELs the arm-test session.

```
ctys \
 -t qemu \
 -a cancel=p:$HOME/qemu/tst/arm-test/arm-test.ctys,\
   force,poweroff:0\
 lab00
```

22.1.5 LIST sessions

The following call LISTs all sessions:

```
TCP-container|TCP-guest  |label   |sesstype|c|user|group
-------------+-----------+--------+--------+-+----+---------
lab00.soho   |-          |arm-test|VNC     |C|acue|ldapusers
lab00.soho   |-          |LAB00   |VNC     |C|root|root
lab00.soho   |-          |LAB00   |VNC     |S|root|root
lab00.soho   |tst109     |arm-test|QEMU-arm|S|acue|ldapusers
lab00.soho   |-          |Domain-0|XEN     |S|-   |-
lab00.soho   |lab00.soho.|lab00   |PM      |S|-   |-
```

The following call LISTs all sessions by usage of a specific LIST-MACRO for QEMU:

```
ctys macro:listconnpid lab00
```

Resulting in:

```
label   |stype    |c|DIS|cport|sport|pid  |PM         |TCP
--------+--------+-+---+-----+-----+-----+----------+------------
LAB00   |VNC      |C|1  |     |     |18933|lab00.soho|
LAB00   |VNC      |S|1  |5901 |     |5642 |lab00.soho|
arm-test|QEMU-arm|S|17 |     |25704|25832|lab00.soho|
Domain-0|XEN      |S|   |     |     |     |lab00.soho|
lab00   |PM       |S|   |     |     |1    |lab00.soho|192.168.1.71
```

22.1.6 ENUMERATE sessions

The following call ENUMERATEs all stored session configurations within the subdirectory of the HOME.

```
ctys -t qemu -a enumerate=b:qemu/tst lab00
```

The following call displays a listing formatted as a table:

```
ctys -t qemu macro:listconn lab00
```

22.1.7 SHOW

The following call SHOWs dynamic data.

```
ctys -t qemu -a show lab00
```

22.1.8 INFO

The following call display static data. Particularly the available capabilities for QEMU are displayed, which contains a list of all available CPUs and the related system boards.

```
ctys -t qemu -a info lab00
```

This leads to:

```
############################
Node:lab00.soho  -  ctys(01_04_001A03)
   System      :Linux
   OS          :Linux
   RELEASE     :2.6.18-8.1.15.el5.centos.plusxen
   MACHINE     :x86_64
   KERNEL#CPU  :SMP-KERNEL
   CPU-INFO
     processor:0
         vendor_id           :GenuineIntel
         cpu family          :6
         model               :22
         model name          :Intel(R) Celeron(R) CPU 420  @ 1.60GHz
         stepping            :1
         cpu MHz             :1599.853
         cache size          :512 KB

      Flags assumed equal for all processors on same machine:
         flags
            vmx(VT-x - Pacifica)    = 0
            svm(AMD-V - Vanderpool) = 0
            PAE                     = 1

   MEM-INFO
      MemTotal               :   523 M
      SwapTotal              :  2031 M

   SOFTWARE
      Mandatory:
         bash        :GNU bash, version 3.1.17(1)-release
                        (x86_64-redhat-linux-gnu)
         gawk        :GNU Awk 3.1.5
         sed         :GNU sed version 4.1.5
         SSH         :OpenSSH_4.3p2, OpenSSL 0.9.8b 04 May 2006
         top         :top: procps version 3.2.7

      Optional:
         wmctrl      :wmctrl is on this machine not available
         lm_sensors  :sensors version 2.10.0 with libsensors version
                      2.10.0
         hddtemp     :hddtemp version 0.3-beta13

   PLUGINS     :QEMU CLI X11 VNC
      QEMU:         Plugin Version:01.01.001a00pre
                    Operational State:ENABLED
                    QEMU version:
                    ->QEMU-0.9.1
                       Magic-Number:QEMU_091
                       Verified Prerequisites:
                       ->CLI-ValidatedBy(hookInfoCheckPKG)
                       ->X11-ValidatedBy(hookInfoCheckPKG)
                       ->VNC-ValidatedBy(hookInfoCheckPKG)
                       -><LocalClientCLI>
                       -><LocalClientX11>
                       -><LocalClientVNC>
                       -><LocalXserverDISPLAY>
                       -><delayedValidationOnFinalTarget>
```

```
                    -><QEMU-0.9.1>
                    ->_/usr/local/bin/vde_switch_info-USER=
                          acue-ACCESS-PERMISSION-GRANTED
                    ->_/usr/local/bin/unixterm_info-USER=
                          acue-ACCESS-PERMISSION-GRANTED
                    ->_/usr/local/bin/vdeq_info-USER=
                          acue-ACCESS-PERMISSION-GRANTED
                    ->_/usr/local/bin/vdeqemu_info-USER=
                          acue-ACCESS-PERMISSION-GRANTED
                    -><QEMUSOCK=/var/tmp/vde_switch0.acue_info-USER=
                          acue-ACCESS-GRANTED>
                    -><QEMUMGMT=/var/tmp/vde_mgmt0.acue_info-USER=
                          acue-ACCESS-GRANTED>
                    -><CPU-Emulation:qemu-alpha>
                    -><CPU-Emulation:qemu-arm>
                    -><CPU-Emulation:qemu-armeb>
                    -><CPU-Emulation:qemu-cris>
                    -><CPU-Emulation:qemu-i386>
                    -><CPU-Emulation:qemu-img>
                    -><CPU-Emulation:qemu-m68k>
                    -><CPU-Emulation:qemu-mips>
                    -><CPU-Emulation:qemu-mipsel>
                    -><CPU-Emulation:qemu-ppc>
                    -><CPU-Emulation:qemu-ppc64>
                    -><CPU-Emulation:qemu-ppc64abi32>
                    -><CPU-Emulation:qemu-sh4>
                    -><CPU-Emulation:qemu-sh4eb>
                    -><CPU-Emulation:qemu-sparc>
                    -><CPU-Emulation:qemu-sparc32plus>
                    -><CPU-Emulation:qemu-sparc64>
                    -><CPU-Emulation:qemu-system-arm>
                    -><CPU-Emulation:qemu-system-cris>
                    -><CPU-Emulation:qemu-system-m68k>
                    -><CPU-Emulation:qemu-system-mips>
                    -><CPU-Emulation:qemu-system-mips64>
                    -><CPU-Emulation:qemu-system-mips64el>
                    -><CPU-Emulation:qemu-system-mipsel>
                    -><CPU-Emulation:qemu-system-ppc>
                    -><CPU-Emulation:qemu-system-ppc64>
                    -><CPU-Emulation:qemu-system-ppcemb>
                    -><CPU-Emulation:qemu-system-sh4>
                    -><CPU-Emulation:qemu-system-sh4eb>
                    -><CPU-Emulation:qemu-system-sparc>
                    -><CPU-Emulation:qemu-system-x86_64>
                    -><CPU-Emulation:qemu-x86_64>

               CLI:         Plugin Version:01.01.001a02
                            Operational State:DISABLED

               X11:         Plugin Version01.01.001a02
                            Operational State:DISABLED

               VNC:         Plugin Version:01.02.001b01
                            Operational State:DISABLED
```

22.1.9 Example Appliances

Qemu Test and Demo Examples

ARM
After installation of QEMU and VDE as described in Section 22.1.1 'Installation of QEMU' on page 445 for validation of the operational state of QEMU the utility "ctys-plugins" should be called. The following call verifies the different plugins operational states for server functionality.

```
ctys-plugins -T all -e
```

The client functionaity could be verified with the call:

```
ctys-plugins -T all
```

Now, with a properly installed test environment from QEMU and the additional ctys call-scripts setup as described before, the following call should start the "arm-test" QEMU VM with and CONSOLE of type SDL.

```
ctys \
 -t qemu \
 -a create=f:qemu/tst/arm-test/arm-test.ctys,\
   console:sdl \
 lab00
```

In case of ambiguous filenames in the cacheDB e.g. due to multiple access paths on multiple nodes by NFS the following approaches could be applied

- use "p:<pathname>"
  When the full absolute path by "p:<pathname>" is provided, no local ambiguity may occur within the execution context.

  This is recommended for the first steps, because it does not require any additional action.

- extend number of params from <machine-address>
  Additional entries may lead to unambiguity. This depends on the contents of the distributed caches and requires some deeper knowledge of the system. Even though this is not that much, it should be shifted when doing the first steps.
- deactivate cacheDB
  Anothe quick solution is the disabling of any caching,

therefore the options "-c off" and "-C off" could be set. This leads to a filesystem scan, which of course results in some performance degradation, which could be serious in case of deep filestructures with a "late match". The scan is performed by usage of the system utility "find".

The supported CONSOLE types for the from-the-box "arm-test" VM are CLI, SDL, and VNC. Additional information is available as inline comment within the "arm-test.ctys" configuration from the

"$HOME/ctys/templates"

directory.

After this call an SDL terminal window should be opened.

In case of networking problems the most common error is the forgotten call of ctys-setupVDE -u <USER> create".

Coldfire

```
ctys \
 -t qemu \
 -a create=f:/qemu/tst/coldfire-test-0.1/coldfire.ctys,\
   console:cli\
 lab00
```

22.2 VMW Examples

22.2.1 Installation of VMW

The Installation of VMware is quite forward. On Systems not supported from the box, the any-any-patch might help.

Some minor pitfalls occur for specific configurations.

- NIC-bonding - bridged
  When a bonding device is utilized on Linux, the mode

6, which is the ARP-negotiation of client and server machines is not supported. The success of the support could be easily checked when using a guest system and calling ping. The effect is the lost of about the half of the ping-answers. This is somewhat a pity, because the mode-6 seems to be the fastest mode which even does not require support intermediate network equipment.

Any other mode seems to work properly.

22.2.2 Installation of Guests

PXE-Boot

The folowing steps are applied to an installation by PXE. This anyhow requires the proper setup of DHCP, TFTP, and one of HTTP/FTP/NFS. For some OSs a so calle "kickstart" file could be used to automate the whole procedure.

For Linux and BSD refer to [133, SYSLINUX] and [134, ETHERBOOT].

(a) Create a VM by native means of a VMware product, but do not start it. When the base machine is created, close the VM.

The common convention within ctys is, that the following items are all literally the same.

- LABEL
- DisplayName
- <vmx-filenameprefix>
- <directory-containing-vmx>

(b) Edit the VMX file manually and apply following changes and addons:

- Check "displayName" as mentioned before.
- Change the ethernet interface entries for the MAC-address and behaviour as required for PXE/DHCP.

  The default behaviour is described as "generatedAddress", which could change and is somewhat chal-

lenging to be continuously maintained for PXE/DHCP. Therefore it should be changed to "static". The resulting entries depend on the actual product, but the essential entries seem to be common as for following example:
- ethernet0.present = "TRUE"
- ethernet0.addressType = "static"
- ethernet0.address = "00:50:56:13:11:4D"

When the cache database is already populated by values from "/etc/dhcpd.conf" or a manually created database similar to "/etc/ethers", the utility ctys-macmap could be used for management of address-mappings.

- Change the UUID entries from dynamic behaviour to static values, otherwise they will change when the machine is reallocated. The values could be kept as already present, else should be generated by "uuidgen".
  - uuid.action = "keep"
  - uuid.location = "56 4d 66 ff 5a 76 d1 19-35 11 73 3d 0f 8d 26 9a"
  - uuid.bios = "56 4d 5e 88 71 0e a5 79-59 6c 34 15 44 a7 7e 96"
- Add ctys-meta information as required for ENUMERATE. Additional values might be applied to the following example.

  The values are not recognized by VMware, thus has to be kept synchronous by the user. The main intention is to get cacheable information for off-line guests to be utilized by ENUMERATE and therefore by "ctys-vhost"

```
#@#IP0="192.168.1.235"
#@#DIST="CentOS"
#@#DISTREL="5"
#@#OS="Linux"
#@#OSREL="2.6"
#@#SERNR="20080415051600"
```

```
#@#CATEGORY="VM"
```

(c) Close the file within the editor and open it again within the VMware frontend.

(d) From now on the following steps could be already proceeded either by native call of "vmware", or by usage of the UnifiedSessionsManager, which has some advances for monitoring and LIST of current active sessions.

The following example illustrates the call, when for a new machine the <machine-adress> is not yet registered within the cacheDB("ctys-vdbgen")

```
ctys -t vmw \
 -a create=\
   p:$HOME/vmware/tst-ctys/tst116/tst116.vmx,\
   reuse\
 -c off  \
 -C off  \
 host1
```

Start the VM and set the emulated BIOS appropriately, by entering with "F12".

(e) Boot into PXE, which might be a simple or more advanced Menu, or just a command line for entering a boot string [133, SYSLINUX] .

(f) Install by means of current GuestOS and/or any generic installer.

ISO-Image and DHCP

The install procedure for usage of an ISO-Image is almos the same as for PXE, just a few formal differences apply.

- Add a CD/DVD-ROM-drive with a link to an ISO-Image, similar to an actual physical drive.
- Select the appropriate boot order, where instead or in addition to PXE the CD/DVD-drive has to be registered.

The required protocols my vary little more, of course.

Install procedures

General Remarks
The most of the installation is performed with PXE if possible. Therefore the PXELINUX is switched to version 3.6.2 and a menu system for the management of the whole test-environment is setup.

When difficulties occur due to specific network requirement, the CD-mount on ISO files option is used, which is almost in any case experienced to be quite safe for VMware products.

CentOS
Installation is in general quite straight-forward. Here performed by PXE on a i386 machine with one CPU.

Debian-4.0_r3
Installation is quite straight-forward, once the required additional netboot-image is downloaded and in place. The differences and additions to the predefined environment are:

(a) Only the kernel image "linux" and the ramdisk "initrd.gz" are imported into a common boot environment with several UNIX variants as provided by [133, SYSLINUX].

(b) The following key has probably to be applied
"debian-installer/allow_unauthenticated=true"

Fedora 8
Installation is quite straight-forward and very similar to CentOS.

MS-Windows-NT-4.0-S
Installed from ISO image.

MS-Windows-2000-WS
Installed from ISO image.

OpenBSD
Installation is quite straight-forward, just the two-level boot has to be considered.

SuSE
Installation is quite straight-forward.

Installed Systems

OS	name	Inst-VM	Media
CentOS-5.0	tst117	Server-1.0.4	PXE
CentOS-5.1	tst112	Server-1.0.4	PXE
Debian-4.0r3	tst106	Server-1.0.4	PXE
Fedora 8	tst103	Server-1.0.4	PXE
OpenBSD-[2-4]	tstXYZ	WS5/6,Server-1.0.[1-5]	PXE,ISO
OpenBSD-4.0	tst109	Server-1.0.4	PXE
OpenBSD-4.2	tst111	Server-1.0.4	PXE
OpenBSD-4.3	tst155	Server-1.0.4	PXE
SuSE-9.3	tstXYZ	WS4/5/6,Server-1.0.[1-5]	PXE,ISO
SuSE-9.3	tst003	WS6,Server-1.0.[345]	PXE,ISO
Solaris 10	tst115	WS6	ISO
Ubuntu-6.06.1-D	tst128	Server-1.0.4	ISO
Ubuntu-6.06.1-S	tst120	Server-1.0.4	PXE
Ubuntu-7.10-S	tst005	Server-1.0.4	PXE
Ubuntu-8.04-D	tst132	Server-1.0.4	ISO
Ubuntu-8.04-S	tst133	Server-1.0.4	PXE
MS-Windows-2000WS	tstXYZ	WS4/5/6,Server-1.0.[1-5]	PXE,ISO
MS-Windows-2000S	tstXYZ	WS4/5/6,Server-1.0.[1-5]	PXE,ISO
MS-Windows-NT-4.0S	tstXYZ	WS4/5/6,Server-1.0.[1-5]	PXE,ISO

Table 22.2: Overview of Intsalled-VMs

22.2.3 CREATE a session

The following call starts a session:

```
ctys \
 -t vmw  \
 -a create=f:vmware/tst-ctys/tst117/tst117.vmx,reuse\
 app2
```

The previous call contains two specifics to be mentioned.

First the filename option "f:" is used, which does a string comparison against the scanned absolute filepaths of configurations files available. The evaluation could be pro-

cessed from cacheDB and/or from the native filesystem on the execution target.

Due to specific handling of filenames just by pattern matching the following call leads to the same result, if unambiguous of course:

```
ctys \
 -t vmw  \
 -a create=f:vmware/tst-ctys/tst117,reuse \
 app2
```

If this is ambiguous, e.g. due to an backup directory, the following could be used too and might solve the problem:

```
ctys \
 -t vmw  \
 -a create=f:vmware/tst-ctys/tst117/t,reuse \
 app2
```

The second part to be mentioned is the "reuse" flag, which initiates simply as first trial a "connect", when this fails, the VM session is created. Thus using the "reuse" flag can lead to some smart handling of sessions, where it is no longer required to remmember whether a session is already present or not. Therefore of course the appropriate configuration of the VM for headless background mode is required.

Another specific case is the usage of a VNCviewer session for a Workstation of Version-6 or later(?). The access requires to be configured by a static port as described within the VMware product manual. The UnifiedSessionsManager provides access by usage of the <machine-address> only, because it has the knowledge how to mat for example the LABEL to a stored vncport. The following example shows a simple redundant access to the proprietary VMware console "CONSOLE:VMW" and the access to "CONSOLE:VNC". Anyhow, the current version of ctys supports only the enumeration of one console for each call.

```
ctys \
 -t vmw  \
 -a create=l:tst117,console:vnc,connect\
 app2
```

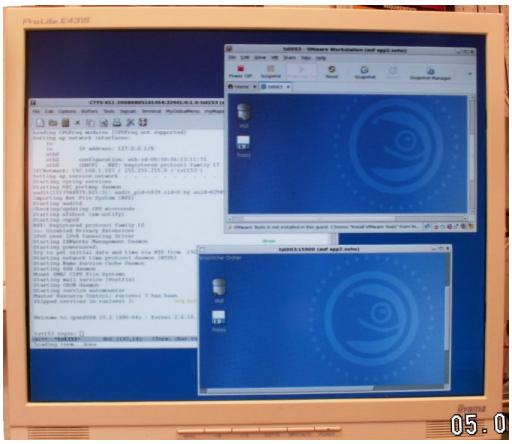

Figure 22.8: VMware WS6 with an additional VNCviewer Client Session.

22.2.4 CANCEL a session

The CANCEL behaviour could be widely configured for VMW. It is e.g. possible to configure an automatic close of the VM, once the GuetsOS is shutdown, when the last VM is stopped, the frontend closes too. This coule be provided by command line options of VMware and is configured as default behaviour for the UnifiedSessionsManager.

The following call CANCELs the VMW without additional user interaction, thus any number of disconnected headless servers could be CANCELed too.

The UnifiedSessionsManager implements the standard behaviour, first to try a native call to the GuestOS, if that fails or a timeout is hit, than the VMware hypervidor interface "vmrun" is called.

```
ctys \
 -t vmw  \
 -a cancel=f:vmware/tst-ctys/tst117/t,poweroff:0 \
 app2
```

Additionaly variants are similar to the provided examples for XEN, this is particularly the case, because they share the most of the top-level CANCEL code.

22.2.5 LIST sessions

The simple LIST call

`ctys -a list app2`

produces the output:

```
TCP-container|TCP-guest        |label  |sesstype|c|user|group
-------------+-----------------+-------+--------+-+----+---------
ws2.soho     |-                |tst100 |VNC     |C|acue|ldapusers
ws2.soho     |ws2.soho.        |ws2    |PM      |S|-   |-
ws2.soho     |-                |tst100 |SSH(XEN)|T|acue|ldapusers
app2.soho    |-                |APP2   |VNC     |C|root|root
app2.soho    |-                |APP2   |VNC     |S|root|root
app2.soho    |tst118           |tst117 |VMW     |S|acue|ldapusers
app2.soho    |tst113           |tst112 |VMW     |S|acue|ldapusers
app2.soho    |tst118           |tst117 |VMW     |C|acue|ldapusers
app2.soho    |tst113           |tst112 |VMW     |C|acue|ldapusers
app2.soho    |app2.soho.       |app2   |PM      |S|-   |-
app2.soho    |00:E0:81:2B:A1:F2|app2   |PM      |S|-   |-
```

This is the default case for two VMs running on app2 with DISPLAYFORWARDING to ws2, and "still" runnng a local client of CLIENTFORWARDING tests for the XEN plugin. The clients and servers for VMW are now coallocated on the server app2.

The CONNECTIONFORWARDING mode is currently supported for:

`Client and Server on different machines:`

```
CONNECTIONFORWARDING
-> Workstation 6+ with VNC client
-> Server with CONSOLE
```

`Client and Server on same machine:`

```
DISPLAYFORWARDING
-> Workstation 6+ with CONSOLE
-> Workstation 6+ with VNC client
-> Server with CONSOLE
```

Thus the following call starts a native frontend with CONNECTIONFORWARDING on server 1.0.4 version:

```
ctys \
   -t vmw \
   -a create=f:vmware/tst-ctys/tst112/t,reuse \
   -L CF \
   olymp
```

The specifics for VMW is, that for the headless-mode initially a complete set with display forwarding is started onm the remote host. ctys starts additionally a local client attached to the configured remote port(default=904) by an encrypted tunnel. The startup of the local client requires in this version an interactive user and password. As far as currently known this has to be a valid local user, a kerberos user seem snot to work. Anyhow, for test purposes here the user "root" was used, which should not be done for productive purposes.

The following list call displays now the complete set of interconnected sessions, for completeness the XEN examples are included in the output.

```
ctys -a list localhost app2 olymp lab00
```

The following listing shows the two clients connected by CONNECTIONFORWARDING, which are a vncviewer connecting as a XEN console to tst100, and a proprietary frontend of VMW connecting to tst112. Both are interconnected by usage of a SSH tunnel implicitly created by the CORE plugin DIGGER and listed as the session type SSH(XEN) and SSH(VMW).

```
TCP-container|TCP-guest         |label   |sesstype|c|user|group
-------------+-----------------+--------+--------+-+----+----------
ws2.soho     |-                 |tst100  |VNC     |C|acue|ldapusers
ws2.soho     |tst112            |tst112  |VMW     |C|acue|ldapusers
ws2.soho     |ws2.soho.         |ws2     |PM      |S|-   |-
ws2.soho     |-                 |tst100  |SSH(XEN)|T|acue|ldapusers
ws2.soho     |-                 |tst112  |SSH(VMW)|T|acue|ldapusers
app2.soho    |-                 |APP2    |VNC     |C|root|root
app2.soho    |-                 |APP2    |VNC     |S|root|root
app2.soho    |tst118            |tst117  |VMW     |S|acue|ldapusers
app2.soho    |tst118            |tst117  |VMW     |C|acue|ldapusers
app2.soho    |app2.soho.        |app2    |PM      |S|-   |-
app2.soho    |00:E0:81:2B:A1:F2|app2    |PM      |S|-   |-
olymp.soho   |tst112            |tst112  |VMW     |S|acue|ldapusers
olymp.soho   |tst112            |tst112  |VMW     |C|acue|ldapusers
olymp.soho   |olymp.soho.       |olymp   |PM      |S|-   |-
lab00.soho   |-                 |tst101  |VNC     |C|acue|ldapusers
lab00.soho   |-                 |LAB00   |VNC     |C|root|root
lab00.soho   |-                 |LAB00   |VNC     |S|root|root
lab00.soho   |-                 |Domain-0|XEN     |S|-   |-
lab00.soho   |tst100            |tst100  |XEN     |S|-   |-
lab00.soho   |tst101            |tst101  |XEN     |S|-   |-
lab00.soho   |lab00.soho.       |lab00   |PM      |S|-   |-
```

22.2.6 ENUMERATE sessions

The following call displays the communications interfaces of the test-pool VMs. For additional information refer to Section 22.2.9 'Display of Available Sessions' on page 475 .

```
ctys -a enumerate=macro:TAB_CPORT,b:vmware/tst-ctys
```

```
Label |stype|cport|PM      |MAC               |TCP
------+-----+-----+--------+------------------+---------------
tst117|VMW  |     |ws2.soho|00:50:56:13:11:52 |192.168.1.240
tst115|VMW  |0    |ws2.soho|00:50:56:13:11:50 |192.168.1.235
tst117|VMW  |     |ws2.soho|00:50:56:13:11:52 |192.168.1.240
tst112|VMW  |     |ws2.soho|00:50:56:13:11:4D |192.168.1.235
tst003|VMW  |0    |ws2.soho|00:50:56:13:11:33 |192.168.1.133
tst005|VMW  |0    |ws2.soho|00:50:56:13:11:35 |192.168.1.135
tst103|VMW  |0    |ws2.soho|00:50:56:13:11:44 |192.168.1.223
tst106|VMW  |0    |ws2.soho|00:50:56:13:11:47 |192.168.1.226
tst111|VMW  |0    |ws2.soho|00:50:56:13:11:4C |192.168.1.234
tst120|VMW  |0    |ws2.soho|00:50:56:13:11:55 |192.168.1.208
tst128|VMW  |0    |ws2.soho|00:50:56:13:11:5C |192.168.1.212
tst002|VMW  |0    |ws2.soho|00:50:56:13:11:32 |192.168.1.132
tst111|VMW  |0    |ws2.soho|00:50:56:13:11:4C |192.168.1.234
```

22.2.7 SHOW

Same as for XEN, Section 22.3.7 'SHOW' on page 499 .

22.2.8 INFO

Same as for XEN, Section 22.3.7 'SHOW' on page 499 .

22.2.9 Display of Available Sessions

Once the basic installation and setup is accomplished, first a "PATHNAME/PNAME" based start of a VM should be performed. The option "-c off" deactivates the use of the nameservice cache for an initially empty cacheDB, thus suppresses several warnings and error messages of internally called tools.

The nest step - after successful installation and configuration of the UnifiedSesssionsManager and a populated cacheDB by usage of "ctys-vdbgen" - might be to list the actually available instances.

The following call of "ctys-vhost" lists all available VMs with given constraints, in this case all instance of VMW which could be started by the user "acue" on the host "app2". The set displayed has to be additionally of the set "tst-ctys", which is the testpool for the UnifiedSessionsManager.

```
ctys-vhost -o pm,label,ids app2 vmw acue tst-ctys
```

The "pm", the "ids" and the "label" are displayed as result.

The additional string "app2 vmw acue tst-ctys" is used as a awk-regexpr and is evaluated as an AND based filter for each word. The whole query requires in this case about 1.4seconds and the following result is displayed.

```
app2.soho;tst117;/homen/acue/vmware/tst-ctys/tst117/tst117.vmx
app2.soho;tst115;/homen/acue/vmware/tst-ctys/tst115/tst115.vmx
app2.soho;tst117;/homen/acue/vmware/tst-ctys/tst117.centos/tst117.vmx
app2.soho;tst111;/homen/acue/vmware/tst-ctys/tst111.OpenBSD-4.2/tst111.vmx
```

Custom Tables

The previous output, which is by default displayed in TERSE format could be formatted by a generic custom table.

The following call displays the required canonical field indexes.

```
ctys-vhost \
 -o pm,label,ids,titleidx \
 app2 vmw acue tst-ctys
```

The indexes are prefixes as an extended table title by "TITLEIDX" .

```
ContainingMachine(1);Label(3);ID(4)
app2.soho;tst117;/homen/acue/vmware/tst-ctys/tst117/tst117.vmx
app2.soho;tst115;/homen/acue/vmware/tst-ctys/tst115/tst115.vmx
app2.soho;tst117;/homen/acue/vmware/tst-ctys/tst117.centos/tst117.vmx
app2.soho;tst111;/homen/acue/vmware/tst-ctys/tst111.OpenBSD-4.2/tst111.vmx
```

This values could be now used to define the output table as:

```
ctys-vhost \
 -o pm,label,ids,tab_gen:1_PM_7%%3_label_4%%4_ID_30 \
 app2 vmw acue tst-ctys
```

As could be seen in the following output, this table configuration is not really helpful. The field sizes are too short, and the common leading part of the pathnames for the ID fields is quite long.

```
    PM     |labe|ID
    -------+----+------------------------------
    app2.so|tst1|/homen/acue/vmware/tst-ctys/ts
    app2.so|tst1|/homen/acue/vmware/tst-ctys/ts
    app2.so|tst1|/homen/acue/vmware/tst-ctys/ts
    app2.so|tst1|/homen/acue/vmware/tst-ctys/ts
```

The following changes might help in advance of usability:

```
ctys-vhost \
 -o pm,label,ids,tab_gen:1_PM_11%%3_label_9%%4_ID_30_L \
 app2 vmw acue tst-ctys
```

Although this is much more helpful, the raise of the ID value should help some more.

```
    PM         |label    |ID
    -----------+---------+------------------------------
    app2.soho  |tst117   |are/tst-ctys/tst117/tst117.vmx
    app2.soho  |tst115   |are/tst-ctys/tst115/tst115.vmx
    app2.soho  |tst117   |-ctys/tst117.centos/tst117.vmx
    app2.soho  |tst111   |/tst111.OpenBSD-4.2/tst111.vmx
```

Thus the final trial for usage and probably storage as a predefined MACRO is:

```
ctys-vhost \
 -o pm,label,ids,tab_gen:1_PM_11%%3_label_9%%4_ID_50_L \
 app2 vmw acue tst-ctys
```

The final result is:

```
PM          |label    |ID
------------+---------+-------------------------------------------------
app2.soho   |tst117   |/homen/acue/vmware/tst-ctys/tst117/tst117.vmx
app2.soho   |tst115   |/homen/acue/vmware/tst-ctys/tst115/tst115.vmx
app2.soho   |tst117   |omen/acue/vmware/tst-ctys/tst117.centos/tst117.vmx
app2.soho   |tst111   |acue/vmware/tst-ctys/tst111.OpenBSD-4.2/tst111.vmx
```

For getting some additional information on the actual installed distributions within the VMs the following call is used:

```
ctys-vhost \
 -o tab_gen:3_label_9%%11_Distro_15%%12_OS_17%%7_TCP_18 \
 app2 vmw acue tst-ctys
```

The final result is:

```
label  |Distro         |OS           |TCP
-------+---------------+-------------+---------------
tst117 |CentOS-5.0     |Linux-2.6    |192.168.1.240
tst115 |Solaris-10     |Solaris-10   |192.168.1.235
tst117 |CentOS-5.0     |Linux-2.6    |192.168.1.240
tst112 |CentOS-5.0     |Linux-2.6    |192.168.1.235
tst003 |SuSE-9.3       |Linux-2.6    |192.168.1.133
tst005 |Ubuntu-7.10-S  |Linux-2.6    |192.168.1.135
tst103 |Fedora-8       |Linux-2.6    |192.168.1.223
tst106 |Debian-4.0r3   |Linux-2.6    |192.168.1.226
tst111 |OpenBSD-4.2    |OpenBSD-4.2  |192.168.1.234
tst120 |FreeBSD-6.1    |FreeBSD-6.1  |192.168.1.208
tst128 |NetBSD-4.0     |NetBSD-4.0   |192.168.1.212
tst002 |SuSE-9.3       |Linux-2.6    |192.168.1.132
tst111 |OpenBSD-4.2    |OpenBSD-4.2  |192.168.1.234
```

The decision is now to use tst117 as test machine.

MACROs for Custom Tables

The previous examples could be stored as MACROs and called just by their macro name. Several preconfigured macros arre available and could be listed with the utility "ctys-macros" .

22.2.10 Example Appliances

FFS.

22.3 Xen Examples

22.3.1 Installation of Xen/Dom0

The installation of the Dom0 is recommended to be based on a available distribution if not required else. A quite straight forward installation could be set up by CentOS/RHEL-5, with additional updates The used from-the-box kernel is "2.6.18-8.1.15.el5.centos.plusxen". For additional information refer to [74, CENTOS].

22.3.2 Installation of Guests/DomU

Templates

The templates supported with ctys are based on the same environment as most of the examples.

The assumed network services are:

```
- DHCP + DNS
- FTP  + NFS
- TFTP
```

For additional services the following will be helpful:

- PXE/PXELinux
- Kerberos+LDAP+(Samba/SMB)+automount

Based on this the test environment is the CentOS-5 distribution with a yum setup and an environment for kickstart install, which could be selected by PXE-menus.

So in accordance to the PXELinux name resolution convention by MAC/host addresses (see gethostip and ctys-vhost) an easy setup based on type-templates and symbolic links

is possible.

The following files support a basic install, which is performed fully automatic by simple scripts. It is based on NFS and FTP. The principle is based on the excellent CentOS HowTo: "Creating and installing a CentOS 5 domU instance" [115, XENGUESTCENTOS] available at the CentOS website.

Just a few adaptations are done and some scripting is applied.

The only required interaction for creation of a DomU with name "tst100" is:

XMCALL="ksu -q -e "./createVM.sh default tst100.i386.anchor

or

XMCALL="ksu -q -e "./createVM.sh default createVM.sh tst100.x86_64.anchor

The parameter XMCALL defines the way permissions to be set for access to restricted resources of Dom0. The first argument "default" defines the install pattern to be used. The second argument "tst100.<arch>.anchor" defines the predefined values for replacing the aliases within the pattern files.

Some preparations has to be done of course. But these are almost only the definitions of some pathnames and the DomU name. Anything else will be derived from the so called anchor-files, including the generation of the required image file for the virtual disk.

The whole process requires 12-20 minutes in average. Starting from the scratch by one call and ending with a running Xen-DomU with a console.

The VMM - Redhat Virtual Machine Manager will be really helpful for monitoring.

createVM.sh
: Main script, replacing own configuration entries within the templates. It generates the required directories, "$HOME/xen/tst-ctys/<DomUName>", the images and starts the install.

MYVM.anchor
: Anchor template and example.

MYVM.conf
: Configuration example for runtime system.

MYVM-inst.conf
: Configuration example for install system.

MYVM.ctys
: Example entries as generated by "ctys-genmconf"

MYVM.ks
: Kickstart file for first basic install

tst100.i386.anchor
: Example for 32bit-para.

tst101.x86_64.anchor
: Example for 64bit-para.

The two created VMs tst100 and tst101 are the test VMs used to perform the basic tests on the XEN plugin.

Install Procedures

CentOS-5
Straight forward as described in [115, XENGUESTCENTOS]. The present template includes a kickstart file for complete automated install.

Fedora-8
Straight forward as described in [116, XENGUESTFEDORA].

OpenSUSE-10.2
The installation of a SuSE GuestOS on Xen could be easily performed with the receipt given by Gerd Hoffmann [118, GHOFFMANN]. This works on CentOS-5 for Dom0 as verified for DomU-SuSE-10.2-x86_64(available from DVD). Just some specifics for post-install boot has to be considered for CentOS-5 at least.

(a) Prepare the install kernel and ramdisk as decribed in [118, GHOFFMANN] and place them appropriately. You need to copy the following files into one build directory

- script called "suse-prepare-install"
- kernel-xen-2.6.18.2-34.x86_64.rpm(10.2)
- install-initrd-1.0-72.x86_64.rpm(10.2)

Than just call "suse-prepare-install"

(b) Install the system e.g. by using the provided ctys-templates by calling from the directory containing the template and performing the native guest installation. The install image should be larger than about 3GBytes, 4GByte is sufficient for a test install. The allocated RAM should be larger than 512MByte, 1GByte is recommended.

Edit the config part of the install template and adapt the generated kernel and ramdisk filepath and start the script.

```
export XMCALL="ksu -q -e "&&\
  ./createVM.sh default tst153.x86_64.anchor
```

(c) Call the install configuration a second time and boot into the system by presented menu. The post-install-boot otherwise causes missing "gfxmenu" error, which alternatively requires a modification of the "/boot/grub/menu" entry of the installed system before reboot.

Once logged in to the guest os comment-out the gfxmenu-line and add appropriate kernel entries.

(d) Now do the post-install-boot "trick" - don't like to spend more time with hd-device and noot of embedded kernel!!!

Just copy the installed kernel and ramdisk from within your guest to the host system and use them(unchanged) with kernel and ramdisk parameter.

Alternatively you can extract them from the rpm-files.

(e) Edit the runtime configuration file of the Xen bootloader:

- kernel=<path>/vmlinuz-2.6.18.2-34-xen

- ramdisk=<path>/initrd-2.6.18.2-34-xen

(f) Start the VM.

That's it.

OpenSUSE-10.1
The installation of a SuSE GuestOS on Xen could be easily performed with the receipt given by Gerd Hoffmann [118, GHOFFMANN]. This works on CentOS-5 for Dom0 as verified for DomU-SuSE-10.1(available from DVD). It seems not to work with DomU-OpenSUSE-10.3.

(a) Prepare the install kernel and ramdisk as decribed in [118, GHOFFMANN] and place them appropriately. Edit the config part of the install template and adapt the generated kernel and ramdisk filepath.

(b) Call template as:

```
export XMCALL="ksu -q -e "&&\
./createVM.sh default tst153.x86_64.anchor
```

Installed Systems

OS	name	Inst-VM	Dom0	DomU
CentOS-5.0	tst100	Xen-3.0.2-Para	X	X
CentOS-5.1	tst101	Xen-3.0.2-Para	X	X
Fedora 8	tst114	Xen-3.0.2-Para	X	X
SuSE-10.1	tst153	Xen-3.0.2-Para		(X)
SuSE-10.2	tst153	Xen-3.0.2-Para		X

Table 22.3: Overview of Intsalled-Xen-VMs

For additional information refer to installed directory.

"$HOME/ctys/templates"

22.3.3 CREATE a session

CREATE a session with CONSOLE:CLI

This call creates a new session by starting a DomU on the host lab00 and opening a CLI console access with the "-c" option within the caller's shell.

Due to pyGRUB an ANSI capable terminal seems to be required, thus starting it within EMACS "shell" will not work.

```
ctys \
 -t xen  \
 -a create=f:xen/tst-ctys/tst100/tst100.conf,\
    CONSOLE:cli \
 -z 2 \
 -b 0,2 \
 lab00'(-Z KSU)'
```

REMARK:

Once a CLI console is attached from a caling shell, the focus might be released by shutdown of the attached VM only, or closing the window containing the session. In X11 environments a graphical console might be preferred for tests.
Using CONSOLE:NONE for delayed attachement of a console is another option.

The local "-z 2" option forces a PTY to be created in any case by calling "ssh -t -t ...". This avoids the remote "TERM=dumb" causing an error of pyGRUB. This is forced by default.

The local "-b 0,2" option forces a serial and non-background mode for interactive shells. Otherwise the console might not work. This is forced by default.

The Remote option "-Z KSU" raises permission by Kerberos on target machine, which is particularly required for the "xm create ..." call. This has to be set as required and may vary for sudo or native root-permissions.

So, using defaults the required call is:

```
ctys \
 -t xen  \
 -a create=f:xen/tst-ctys/tst100/tst100.conf,\
    CONSOLE:cli \
 lab00'(-Z KSU)'
```

The "CONSOLE:NONE" suboption just creates a server in so called headless-mode.

```
ctys \
 -t xen  \
 -a create=l:tst100,CONSOLE:none \
 lab00'(-Z KSU)'
```

The following calls just connect to a running instance.

In this case the pathname is used.

```
ctys \
 -t xen  \
 -a create=p:$HOME/xen/tst-ctys/tst100/tst100.conf,\
    CONSOLE:cli,connect \
 lab00'(-Z KSU)'
```

As a variation a relative filename for comparison of "find" results could be used in variable length, as long as the match is unambiguous.

```
ctys \
 -t xen  \
 -a create=f:xen/tst-ctys/tst100/tst100.conf,\
    CONSOLE:cli,connect \
 lab00'(-Z KSU)'
```

Same result with:

```
ctys \
 -t xen  \
 -a create=f:tst-ctys/tst100/tst100.conf,\
    CONSOLE:cli,connect \
 lab00'(-Z KSU)'
```

Another call variation:

```
ctys \
 -t xen  \
 -a create=f:tst100/tst100.conf,\
    CONSOLE:cli,connect \
 lab00'(-Z KSU)'
```

In this case the id, which is for Xen "id==pathname", is used.

```
ctys \
 -t xen  \
```

```
 -a create=i:$HOME/xen/tst-ctys/tst100/tst100.conf,\
    CONSOLE:cli,connect \
 lab00'(-Z KSU)'
```

In the next case the label is used.

```
ctys \
 -t xen  \
 -a create=l:tst100,CONSOLE:cli,connect \
 lab00'(-Z KSU)'
```

The following call just connects to a running instance too, but uses the UUID.

```
ctys \
 -t xen  \
 -a create=u:6842caf91e3e43249ed596b8b9f2c5c2,\
    CONSOLE:cli,connect \
 lab00'(-Z KSU)'
```

The same when using MAC address.

```
ctys \
 -t xen  \
 -a create=m:00:50:56:13:11:40,\
    CONSOLE:cli,connect \
 lab00'(-Z KSU)'
```

The same when using IP address.

```
ctys \
 -t xen  \
 -a create=t:192.168.1.220,\
    CONSOLE:cli,connect \
 lab00'(-Z KSU)'
```

CREATE a session with CONSOLE:XTERM

This call creates a new session by starting a DomU on the host lab00 and opening a CLI console access with the "-c" option within a newly created xterm window.

```
ctys \
 -t xen  \
 -a create=f:xen/tst-ctys/tst100/tst100.conf,\
    CONSOLE:xterm \
 lab00'(-Z KSU)'
```

The creation of PTY is not required. The Remote option "-Z KSU" raises permission by Kerberos, which is particularly required for the "xm create ..." call.

CREATE a session with CONSOLE:GTERM

Almost the same as XTERM, but a "gnome-terminal" is created instead.

```
ctys \
 -t xen  \
 -a create=f:xen/tst-ctys/tst100/tst100.conf,\
    CONSOLE:gterm \
 lab00'(-Z KSU)'
```

CREATE a session with CONSOLE:EMACS

The EMACS console starts an EMACS and executes the call within a "shell" buffer named with LABEL of current XEN instance. The "ansi-term" is for now supported only when "ctys" is executed within as native call. The only drawback for the "shell" buffer is the lack of ansi-color support by ctys s and some restrictions due to lack of some ANSI terminal functions.

```
ctys \
 -t xen  \
 -a create=f:xen/tst-ctys/tst100/tst100.conf,\
    CONSOLE:emacs \
 lab00'(-Z KSU)'
```

The following image shows an EMACS variant, where the ANSI-TERM mode is choosen and the window is slitted by the call automatically into two parts. The lower displays and prompt for the <exec-target>, whereas the top-window shows the prompt of a shell access into the GuestOS, which is given by the <machine-address> of the call.

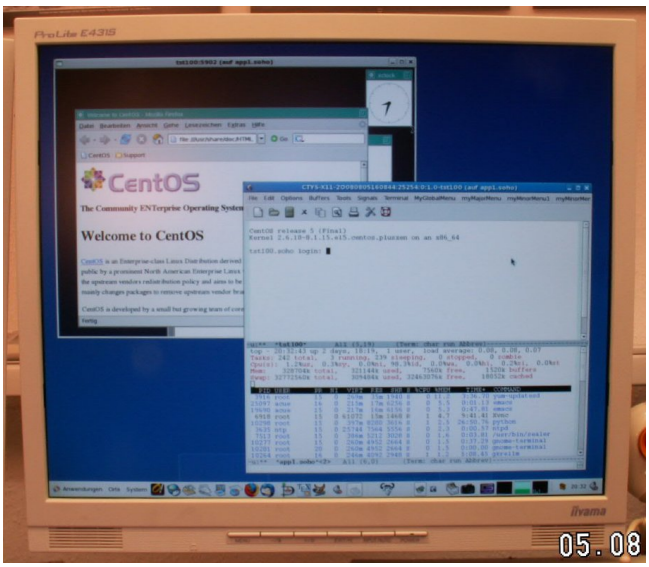

Figure 22.18: The "CONSOLE:EMACSAM" for a XEN Session

CREATE a session with CONSOLE:VNC

This call creates a session with an attached VNC viewer as a console. Therefore it is highly recommended to set the "vncunused=1" value in order to use a free port. When this is set to "vncunused=0" interference with native VNC servers might occur. The complete set of recommended VNC settings are:

```
vnc = 1
vncconsole = 1
vncunused = 1
```

The attachement of the console by a vncviewer is in ctys processed as a seperate step. Due to the asynchronous start of the DomU a timeout is implemented, which delays the start of the VNC console. This value coudl be configured by the user.

The resulting call for starting the session is:

```
ctys \
 -t xen  \
 -a create=f:xen/tst-ctys/tst100/tst100.conf,\
    CONSOLE:vnc \
 lab00'(-Z KSU)'
```

The previous call implies the "-L DF" option for DISPLAYFORWARDING , the same call could be performed with "-L CF" for CONNECTIONFORWARDING .

```
ctys \
 -t xen  \
 -a create=f:xen/tst-ctys/tst100/tst100.conf,\
    CONSOLE:vnc \
 -L CF\
 lab00'(-Z KSU)'
```

Now the advantage of a formal split for Client and Server, where the client is attached by a seperate step, should be clear.

The result could be verified by calling

ctys -a list

on the client machine, which results e.g. to:

```
TCP-container |TCP-guest |label   |sesstype|c|user |group
--------------+----------+--------+--------+-+-----+----------
ws2.soho      |-         |tst100  |VNC     |C|acue |ldapusers
ws2.soho      |ws2.soho. |ws2     |PM      |S|-    |-
ws2.soho      |-         |tst100  |SSH(XEN)|T|acue |ldapusers
```

The previous output is the standard table displayed, but could be completely customized by the user.

The "sesstype" representing the session type "SSH(XEN)" displays the tunnel created by the internal DIGGER plugin and charachterizes it by "T" as a tunnel. The label is here the same as for the the VNC session, which is characterized by "C" as a client, attached to the sessions server, the Xen DomU tst100 on the remote machine lab00. The client(tst100) and server(tst100) are interconnected via the tunnel tst100. For additional customization, e.g. the SORT attribute refer to LIST action.

The following output shows both machines, the localhost as client and the lab00 as the server. The call is varied to

```
ctys -a list localhost lab00
```

and displays:

```
TCP-container |TCP-guest  |label   |sesstype|c|user |group
--------------+-----------+--------+--------+-+-----+----------
ws2.soho      |-          |tst100  |VNC     |C|acue |ldapusers
ws2.soho      |ws2.soho.  |ws2     |PM      |S|-    |-
ws2.soho      |-          |tst100  |SSH(XEN)|T|acue |ldapusers
lab00.soho    |-          |Domain-0|XEN     |S|-    |-
lab00.soho    |tst100     |tst100  |XEN     |S|-    |-
lab00.soho    |lab00.soho.|lab00   |PM      |S|-    |-
```

CREATE a session with CONSOLE:NONE

This call enters so called "headless-mode".

```
ctys \
 -t xen  \
 -a create=f:xen/tst-ctys/tst100/tst100.conf,\
    CONSOLE:none \
 lab00'(-Z KSU)'
```

CREATE a session with RESUME from stat-file

```
ctys \
 -t xen  \
 -a create=l:tst100,RESUME:mystate.stat \
 lab01
```

CREATE a session with RESUME with VNC

```
ctys \
 -t xen  \
 -a create=l:tst100,RESUME,CONSOLE:VNC \
 lab01
```

CREATE a session with RESUME with EMACS

```
ctys \
 -t xen  \
 -a create=l:tst100,RESUME,CONSOLE:EMACS \
 lab01
```

CREATE a multiple sessions

The following call creates two sessions with one call. Both sessions are here located on the physical machine lab00 and use "ksu" for raise of access permissions.

```
ctys \
 -t xen  \
 -- \
 '(-Z KSU)' \
 lab00'(\
    -a create=f:xen/tst-ctys/tst100/tst100.conf,\
       CONSOLE:none\
 )' \
 lab00'(\
    -a create=f:xen/tst-ctys/tst101/tst101.conf,\
       CONSOLE:none\
 )'
```

The same could be varied for example to use different "-Z" options with "KSU" as default:

```
ctys \
 -t xen  \
 -- \
 '(-Z KSU)' \
 lab00'(\
   -a create=f:xen/tst-ctys/tst100/tst100.conf,\
      CONSOLE:none\
  )' \
  lab00'(\
   -a create=f:xen/tst-ctys/tst101/tst101.conf,\
   CONSOLE:none -Z SUDO\
  )'
```

The same could be varied for example to use different "-Z" options with none as default:

```
ctys \
 -t xen  \
 -- \
 lab00'(\
   -a create=f:xen/tst-ctys/tst100/tst100.conf,\
      CONSOLE:none -Z KSU\
 )'
 lab00'(\
```

```
   -a create=f:xen/tst-ctys/tst101/tst101.conf,\
   CONSOLE:none -Z SUDO\
 )'
```

It might be obvious howto use different physical hosts:

```
ctys \
  -t xen  \
  -- \
  lab00'(\
    -a create=f:xen/tst-ctys/tst100/tst100.conf,\
       CONSOLE:none -Z KSU\
  )'
  lab01'(\
    -a create=f:xen/tst-ctys/tst101/tst101.conf,\
       CONSOLE:none -Z SUDO\
  )'
```

22.3.4 CANCEL a session

CANCEL a session with POWEROFF

This call stops the DomU addressed by it's PATHNAME.

```
ctys \
 -t xen \
 -a cancel=poweroff:0,\
    p:/homen/acue/xen/tst-ctys/tst100/tst100.conf\
 lab00'(-Z KSU)'
```

For the following calls caching is used by default, which could lead to errors when ambiguity of addressed targets occur. When ambiguity occurs, additional <machine-address> parts might resolve this.

As a work around for handling multiple copies, such as backups, with identical address contents, one of the following approaches might help:

- The cache could be deactivated by using the options "-c off" and/or "-C off" .
- The usage of a PATHNAME might resolve ambiguity too, when resolved on the target only. Be aware, that this could be naturally ambiguous too in NFS environments with automount, of course. The latter will be frequently the case when configuring a load balancing environment by mounting VM collections.

- The cache could be rebuild by an appropriate selection in combination of a review of the contents of the filesystem.

The actual call is displayed within the trace output and could be called after cut-and-paste to command line. A listing of all actual contained instances with ambiguous addresses is listed by the ctys-vhost option "-M all" .

Same by using the LABEL as address.

```
ctys \
 -t xen \
 -a cancel=l:tst100,POWEROFF:0 \
 lab01
```

When ambiguity occurs, e.g. like depicted by followin example:

```
ctys-vhost -o machine -s -M all lab00 XEN tst100

lab00.xyz;XEN;tst100;\
    /root/xen/tst100/tst100.conf;\
    6842caf91e3e43249ed596b8b9f2c5c2;\
    00:50:56:13:11:40;192.168.1.220;;;;CentOS;Linux;5;;PM

lab00.xyz;XEN;tst100;\
    /root/xen/tst100/tst100-inst.conf;;\
    00:50:56:13:11:40;192.168.1.220;;;;;;;;;

lab00.xyz;XEN;tst100;\
    /homen/chkusr/xen/tst-ctys/tst100/tst100.conf;\
    6842caf91e3e43249ed596b8b9f2c5c2;\
    00:50:56:13:11:40;192.168.1.220;;;;CentOS;Linux;5;\
    20080427002200;VM
```

The following call resolves ambiguity by deactivating cached operations:

```
ctys \
 -t xen \
 -a cancel=l:tst100,POWEROFF:0 \
 -C off \
 lab01
```

Similar with additional deactivation of nameservice caching, which anyhow is used sparsely for LIST action in current version.

```
ctys \
 -t xen \
 -a cancel=l:tst100,POWEROFF:0 \
 -c off \
 -C off \
 lab01
```

CANCEL a session with RESET

Similar call to previous, but reboots after resetting hypervisor. When SELF is selected the hosting machine will be RESET too, else the GuestOS within the hypervisor only.

```
ctys \
 -t xen \
 -a cancel=l:tst100,RESET \
 -c off \
 -C off \
 lab01
```

The following call ignores the eventual contained VMs within a stacked XEN instance. Actually the only VM supported to be executed nested within another is of type QEMU.

```
ctys \
 -t xen \
 -a cancel=l:tst100,FORCE,RESET \
 -c off \
 -C off \
 lab01
```

CANCEL a session with REBOOT

Similar call to previous, but reboots after shutdown.

```
ctys \
 -t xen \
 -a cancel=l:tst100,REBOOT \
 -c off \
 -C off \
 lab01
```

The following call ignores the eventual contained VMs within a stacked XEN instance. Actually the only VM supported to be executed nested within another is of type QEMU.

```
ctys \
 -t xen \
 -a cancel=l:tst100,FORCE,REBOOT \
 -c off \
 -C off \
 lab01
```

Same with pathname, should be used for tests, due it's evaluation means for a missing label.

```
ctys \
 -t xen \
 -a cancel=FORCE,REBOOT,\
    p:/homen/chkusr/xen/tst-ctys/tst100/tst100.conf \
 -c off \
 -C off \
 lab01
```

CANCEL a session with PAUSE

Currently not yet available.

```
ctys \
 -t xen \
 -a cancel=l:tst100,PAUSE \
 lab01
```

```
ctys \
 -t xen \
 -a cancel=l:tst100,FORCE,PAUSE \
 lab01
```

CANCEL a session with SUSPEND

Currently not yet available.

```
ctys \
 -t xen \
 -a cancel=l:tst100,SUSPEND \
 lab01
```

```
ctys \
 -t xen \
 -a cancel=l:tst100,FORCE,SUSPEND \
 lab01
```

CANCEL a session with INIT

Calls UNIX "init" call with provided level. This call is somewhat limited for now, RESET and REBOOT should be preferred.

```
ctys \
 -t xen \
 -a cancel=l:tst100,INIT:0 \
 lab00
```

```
ctys \
 -t xen \
 -a cancel=l:tst100,FORCE,INIT:6 \
 lab00
```

22.3.5 LIST sessions

List sessions.
The following call lists all sessions as MACHINE format raw records, a prefix title with given raw indexes is displayed. The provided indexes are the values to be used to define custom tables to be stored as macros.

```
ctys \
 -t xen \
 -a list=machine,titleidx \
 lab00
```

A simple call with default values displays the standard output.

```
ctys -a list lab01
```

The following result is displayed.

```
TCP-container|TCP-guest  |label   |sesstype|c|user|group
-------------+-----------+--------+--------+-+----+-----
lab00.soho   |-          |LAB00   |VNC     |C|root|root
lab00.soho   |-          |LAB00   |VNC     |S|root|root
lab00.soho   |-          |Domain-0|XEN     |S|-   |-
lab00.soho   |tst100     |tst100  |XEN     |S|-   |-
lab00.soho   |tst101     |tst101  |XEN     |S|-   |-
lab00.soho   |lab00.soho.|lab00   |PM      |S|-   |-
```

When the subsystem XEN is selected the output is reduced to XEN only.

```
ctys -t xen -a list lab01
```

The following result is displayed.

```
TCP-container|TCP-guest  |label   |sesstype|c|user|group
-------------+-----------+--------+--------+-+----+-----
lab00.soho   |-          |Domain-0|XEN     |S|-   |-
lab00.soho   |tst100     |tst100  |XEN     |S|-   |-
lab00.soho   |tst101     |tst101  |XEN     |S|-   |-
```

A running configuration with two XEN sessions, where one session tst101 is connected by DISPLAYFORWARDING and a second tst100 is connected by CONNECTIONFORWARDING is displayed with the call

```
ctys -t xen -a list localhost lab01
```

as follows.

```
TCP-container|TCP-guest  |label   |sesstype|c|user|group
-------------+-----------+--------+--------+-+----+---------
ws2.soho     |-          |tst100  |VNC     |C|acue|ldapusers
ws2.soho     |ws2.soho.  |ws2     |PM      |S|-   |-
ws2.soho     |-          |tst100  |SSH(XEN)|T|acue|ldapusers
lab00.soho   |-          |tst101  |VNC     |C|acue|ldapusers
lab00.soho   |-          |LAB00   |VNC     |C|root|root
lab00.soho   |-          |LAB00   |VNC     |S|root|root
lab00.soho   |-          |Domain-0|XEN     |S|-   |-
lab00.soho   |tst100     |tst100  |XEN     |S|-   |-
lab00.soho   |tst101     |tst101  |XEN     |S|-   |-
lab00.soho   |lab00.soho.|lab00   |PM      |S|-   |-
```

The next call lists all communication related informations by usage of the predefined custom table stored as macro "TAB_CPORT"

```
ctys -a list=macro:TAB_CPORT localhost lab00
```

results to:

```
Label     |stype   |cport|PM         |MAC               |TCP
----------+--------+-----+-----------+------------------+---------------
tst100    |VNC     |     |ws2.soho   |                  |
ws2       |PM      |     |ws2.soho   |00:1D:60:A5:89:06|192.168.1.70
tst100    |SSH(XEN)|5950 |ws2.soho   |                  |
tst101    |VNC     |     |lab00.soho|                  |
LAB00     |VNC     |     |lab00.soho|                  |
LAB00     |VNC     |5901 |lab00.soho|                  |
Domain-0  |XEN     |     |lab00.soho|                  |
tst100    |XEN     |5900 |lab00.soho|00:50:56:13:11:40|
tst101    |XEN     |5902 |lab00.soho|00:50:56:13:11:41|
lab00     |PM      |     |lab00.soho|00:0E:0C:35:F8:48|192.168.1.71
```

Figure 22.24: TAB_CPORT by LIST

As could be recognized, the VMs tst100 and tst101 has no TCP values displayed, even though these are present. The reason is simply the decision to only display data which could be fetched easily unambiguously. The TCP address is only in a simple 1-to-1 relation, when no additional interfaces are present, and when the mapping information of the actual TCP stack and the ctys configuration including it's cacheDB are consistent. Additionally all services has to be setup properly, e.g. when using "host" or "dig". Another point is that the VM has to be connected to the managing nameservices. Thus the complete automatic implementation is somewaht advanced and is shifted for now. In current version the user has to poll the missing information by additional tools, such as ctys-vhost, ctys-macmap, or ctys-dnsutil, or simply by "host" or "dig."

Anyhow, the ENUMERATE action displays the TCP addresses as they are configured within the configuration file, refer for the output of the same common generic table "TAB_CPORT by ENUMERATE" as an complementary example. Additionally the same table could be used for "ctys-vhost" with a similar result to ENUMERATE: "TAB_CPORT by VHOST" .

22.3.6 ENUMERATE sessions

The following call enumerates all VMs

```
ctys \
```

```
 -t xen \
 -a enumerate=machine,title,b:xen \
 localhost lab00
```

The complementary example for the common generic table "TAB_CPORT by LIST" could be generated by the call

```
ctys \
 -a enumerate=macro:TAB_CPORT,b:xen%/etc/ctys.d
 localhost lab00
```

and displays some of the basic differences in the output strategy. As the following output depicts, here the fields for the TCP address are filled, whereas no cport is displayed.

The TCP addresses are ere displayed as statically configured within the configuration file. The cport is the communications port for the client processes, in this case the VNC port, which is dynamically allocated due to preconfigured "vncunused=1". Thus the value is defined during runtime only, so it is not displayed by ENUMERATE, which displays the statically configured data.

```
Label |stype|cport|PM        |MAC              |TCP
------+-----+-----+----------+-----------------+-------------
tst100|XEN  |     |ws2.soho  |00:50:56:13:11:40|192.168.1.220
tst101|XEN  |     |ws2.soho  |00:50:56:13:11:41|192.168.1.221
tst104|XEN  |     |ws2.soho  |00:50:56:13:11:44|192.168.1.224
ws2   |PM   |     |ws2.soho  |00:1D:60:A5:89:06|192.168.1.70
tst100|XEN  |     |lab00.soho|00:50:56:13:11:40|192.168.1.220
tst101|XEN  |     |lab00.soho|00:50:56:13:11:41|192.168.1.221
tst104|XEN  |     |lab00.soho|00:50:56:13:11:44|192.168.1.224
lab00 |PM   |     |lab00.soho|00:0E:0C:35:F8:48|192.168.1.71
```

Figure 22.25: TAB_CPORT by ENUMERATE

Additionally the same table could be used for "ctys-vhost" with a similar result to ENUMERATE: "TAB_CPORT by VHOST" .

22.3.7 SHOW

Lists the dynamic global envvironement data on the target.

```
ctys -t xen -a show lab00
```

Same result with

```
ctys -a show lab00
```

22.3.8 INFO

Lists static data for configured UnifiedSessionsManager with configuration relevant resource data.

The following call lists the initialized XEN plugin with implicitly loaded additional uninitialized plugins.

```
ctys -t xen -a info lab01
```

The following call lists all available plugings with their resulting init states on the target.

```
ctys -a info lab01
```

22.3.9 Example Appliances

FFS.

Chapter 23

PMs - Sessions

23.1 PM Examples

23.1.1 Configuration of Access Permissions

The plugin PM accesses several basic system resources, which require by default root permissions. These resources are mainly required for CANCEL of machine and in case of WoL for the startup of a machine by CREATE.

The CREATE action is splitted into two security cases:

- Send WoL on local segment.
- Send WoL to a remote segment

For the packet distribution to the local segment the tool "ethtool" is used which requires root permissions to the interface to be used. For the remote distribution an own script is utilized, which does not require ans specific permissions on the sending host, but some preparation on routers connecting the target segment.

The CANCEL action is only executed on the target machines, but requires on these several system facilities. The minimal requirement are the "shutdown-tools" on machines with non-bridged interfaces. On machines with bridged interfaces without an additional WoL interface some facilities for controlling the bridge are eventually required.

The following access permissions are required for the current release of ctys. For standard machines without required bridge shutdown for setting "wol g":

/sbin/halt
/sbin/reboot
/sbin/poweroff
/sbin/init
/sbin/ethtool
/sbin/ether-wake

For machines with bridges required to be shutdown for setting "wol g", typically Xen-3.0.2, the following additional permissions have to be configured:

/usr/sbin/brctl
/sbin/ip
/sbin/ifup
/sbin/ifdown

For Xen the following tools with root-permissions are required:

/usr/sbin/xm
/usr/bin/virsh

The permissions should be configured as described in the related chapters.

Due to the timeout behaviour of ksu and sudo during probing when no user is configured, the default behaviour is not to probe for theese tools.

Additional information is provided in the example for handling WoL.

23.1.2 CREATE a PM Session

CREATE with WoL or RESUME

This will switch on a machine previously shutdown with WoL. The execution host for ctys tasks is "lab01", so root permission as required has to be granted on "lab01" only.

RESUME is internally mapped to WoL, thus the same.

```
ctys \
  -t pm  \
  -a create=wol,t:hostX,broadcast:hostXDirectedCast \
  lab01
```

CREATE with CONSOLE

This will create a new session to a running machine with a CONSOLE any CLI, XTERM, GTERM, EMACS, or VNC.

```
ctys -t pm -a create=CONSOLE:cli
```

23.1.3 CANCEL a PM Session

CANCEL a session with POWEROFF

This cancels anything running on top of the PM, but by default not the PM itself. Thus the whole VM stack running on top of the PM will be powered off by recursive stack handling.

```
ctys -t pm -a cancel=l:tst100,POWEROFF:0 lab01
```

The suboption FORCE just calls the lowest VM hypervisors for immediate - non-stack - poweroff. Contained upper VMs might not be able to store cached data, thus could be in erroneous state after switching them off abruptly.

```
ctys -t xen -a cancel=l:tst100,force,POWEROFF:0 lab01
```

The suboption WOL without a BROADCAST suboption sends a broadcast packet for WoL into local segment only. Therefore the first interface - which could be a bonding device - is used. When this has to be altered, the broadcast parameter has to be supplied.

```
ctys -t xen -a cancel=l:tst100,POWEROFF,wol lab01
```

The same with a "directed-broadcast".

```
ctys \
 -t xen \
 -a cancel=l:tst100,POWEROFF,wol,\
    broadcast:192.168.3.255 \
 lab01
```

Directed broadcast is preferred here, due to support of the native OpenBSD based routers, and the internal-only usage. This could be seen as "somewhat secure", because of fine-grained and rigorous filtering rules in addition.

CANCEL with REBOOT

This call REBOOTs all VMs running on top of PM, but not the PM itself. In addition a LABEL is supported, which just names the current session for display purposes.

Therefore an appropriate stack-propagation is performed.

```
ctys -t pm -a cancel=l:tst001,REBOOT lab01
```

The following call reboots in addition the PM itself.

```
ctys -t pm -a cancel=l:tst001,REBOOT,SELF lab01
```

This call just reboots the PM, NO STACK-PROPAGATION is performed, thus OSs within upper VMs might be corrupted, at least require some kind of recovery mechanisms.

```
ctys -t xen -a cancel=l:tst100,REBOOT,FORCE lab01
```

CANCEL with PAUSE,SUSPEND

Current version just remaps this to a POWEROFF, therefore the user has to support the only supported Wake-Up mechanism WoL, refer to POWEROFF with WoL.

CANCEL with INIT

INIT is a transparently mapped action, which is almost the same as a native init-call. The difference is the call-relocation to the execution-target machine only.

Therefore the caller is responsible for the match of the requested init level with additional attributes, e.g. a WoL entry might not make too much sense, when called with "init 3".

```
ctys -t PM -a cancel=l:tst100,INIT:0 lab01
```

23.1.4 PM - Using Wake-On-Lan - WoL

The WoL feature is based on the following pre-requirements:

(a) Support of the NIC for WoL, an "active sleep mode"

(b) Support of the Motherboard, namely the BIOS, and if present the OnBoard-NIC for WoL.
Depending from the bus design and/or on-board LAN, e.g. the "PME" feature for the PCI bus is required for NIC cards, and additionaly a WoL feature. Both options are usually located within the Power Management settings of the BIOS.

(c) The pre-shutdown setting of a volatile flag on the NIC, which activates the autonomous "active sleep mode" of the card.

This could be established persistently by adding it to an init script. Alternatively ctys CANCEL action could be used only for "WOL" option, simulating persistency by refreshing the setting on each CANCEL.

The setting is only stored as long as the NIC is in standby mode, e.g. the S5 state of ACPI, thus lost when the Powersupply of the machine is actually switched off. Anyhow, solutions may exist, which store the attribute in a non-volatile storage.

The tools required for ctys are:

- ethtool, for setting the NIC to WoL.
- netcat for sending the WoL packet to remote segments.
- ether-wake for sending the WoL packet to local segment.
- halt, poweroff, shutdown, and reboot for cancel action.

The permissions required are:

- root permission for ethtool for write access.
- root permission for halt, poweroff, shutdown, and reboot, when no wrapper like the consolehelper of CentOS/RHEL is present.

The configuration of ksu and/or sudo should be preferred for granting access permissions, the change of file permissions should be avoided.

Thus the usage of WoL with ctys calls only requires the for example the following calls:

(a) Switch the hostX into stand-by mode.

```
ctys -t pm -a cancel=wol,poweroff hostX
```

(b) Activate the hostX from stand-by mode.

```
ctys -t pm -a create=wol,t:hostX,\
  broadcast:192.168.3.255
```

t:hostX
: Could be any valid <machine-address>, e.g. "m:macX" or "l:labelX".

broadcast:192.168.3.255
: Send a broadcast UDP to port 9 on given address. This has to be supported by the router. When not present a broadcast to the local subnet only is sent.

  Alternatively the final router of target subnet could be configured for redirection of a specific address/port to a local brooadcast.

  For example for "192.168.3.251" on port "4711", the following could be used: "broadcast:192.168.3.251:4711".

Some supported and verified scenarios for WoL application. These scenarios show up one of the main pitfalls within a SSO environment, where e.g. a kerberos based "ksu" does not work for any case.

(a) A single NIC as unique remote access port in a production environment. No bridges are present.

  Anything works quite straight forward. Due to the required root-permissions any shutdown-utility could be used with local and network based users.

```
ctys \
  -t pm \
  -a create=wol,t:<hostX> -Z NOSUDO \
  '(-d 99 -Z KSU)'

ctys \
  -t pm \
```

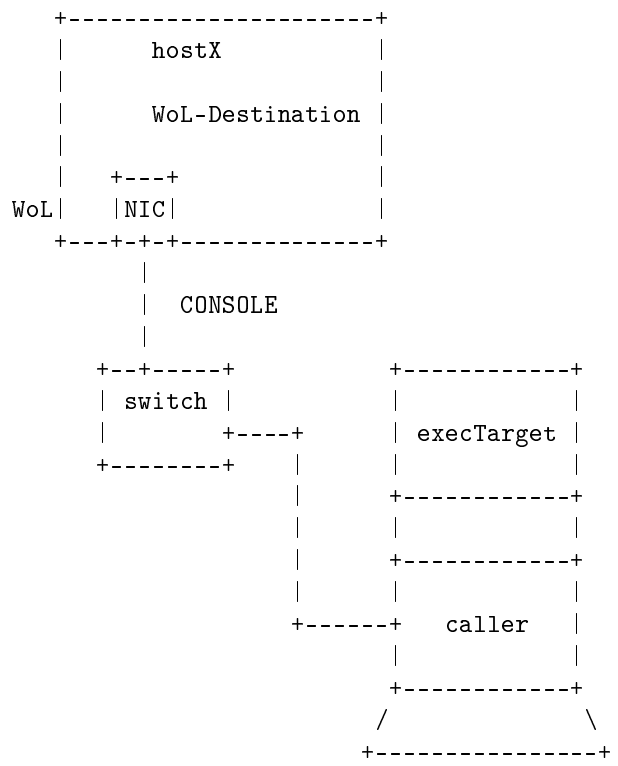

Figure 23.1: "Bridge-less" WoL configuration within same Ethernet Segment

```
-a cancel=wol,self,stack,poweroff:0 \
-Z KSU \
<hostX>'(-d 99 -w -Z SUDO)'
```

(b) A single bridged NIC as unique remote access port in a production environment.

In this scenario things become somewhat complicated. The main reason for now is, that in case of a bridges physical NIC, the "wol g" option of ethtool is not correctly set on the physical nor the logical NICs beeing a member of a bridge.

Thus a "stop" of the bridge is required previously to setting WoL, which - at least temporarily - disconnects "<hostX>" from the network. And therefore from the authentication services as well as from the required runtime-utilities.

Thus this is supported for local users only.

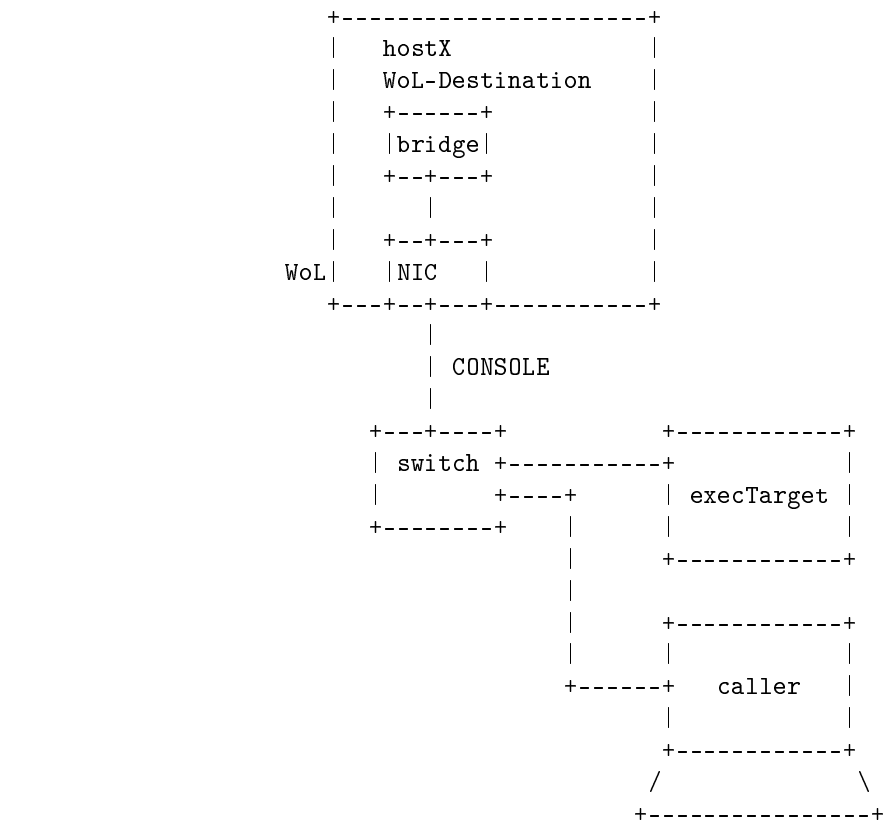

Figure 23.2: WoL configuration with a virtual bridge

```
ctys \
 -t pm \
 -a create=wol,t:<hostX> -Z NOSUDO

ctys \
 -t pm \
 -a cancel=wol,self,stack,poweroff:0 \
 -Z NOKSU \
 <hostX>'(-Z SUDO)'
```

(c) A bridged bonded NIC group as unique remote access port in a production environment.

The same restrictions as for 2.).

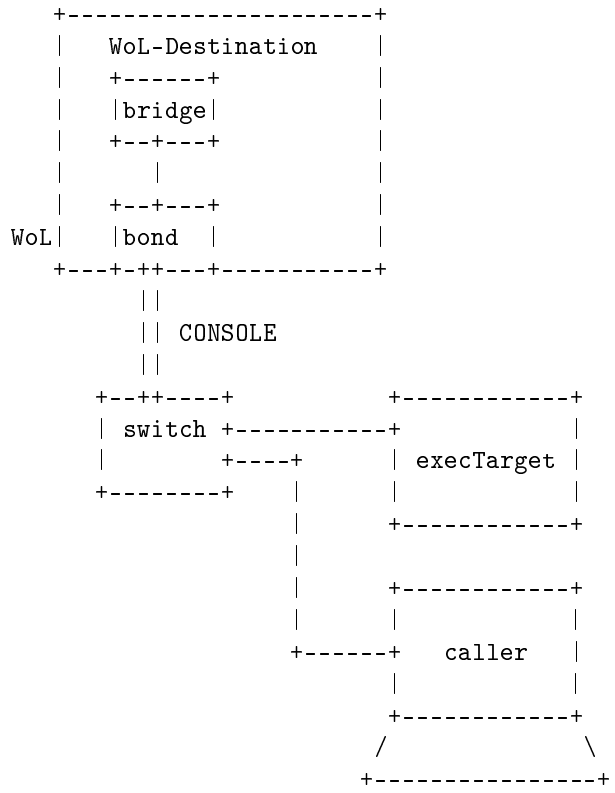

Figure 23.3: WoL configuration with a virtual bridge on a bond device

```
ctys \
 -t pm \
 -a create=wol,t:<hostX> -Z KSU \
 <execTarget>'(-Z SUDO)'

ctys \
 -t pm \
 -a cancel=wol,self,stack,poweroff:0 \
 -Z NOKSU \
 <hostX>'(-Z SUDO)'
```

(d) A bridged bonded NIC group as non-WoL remote access port in a production environment. Additionally a specific management NIC with WoL support.

In this case restrictions to local-only users apply only,

when the WoL-NIC is used as access port. In any case no local restrictions apply to network-users with mounted HOMEs.

But instead some configuration of network equipment is required in order to pass the required messages through involved routers.

A typical scenario where a remote broadcast is required even in the local LAN.

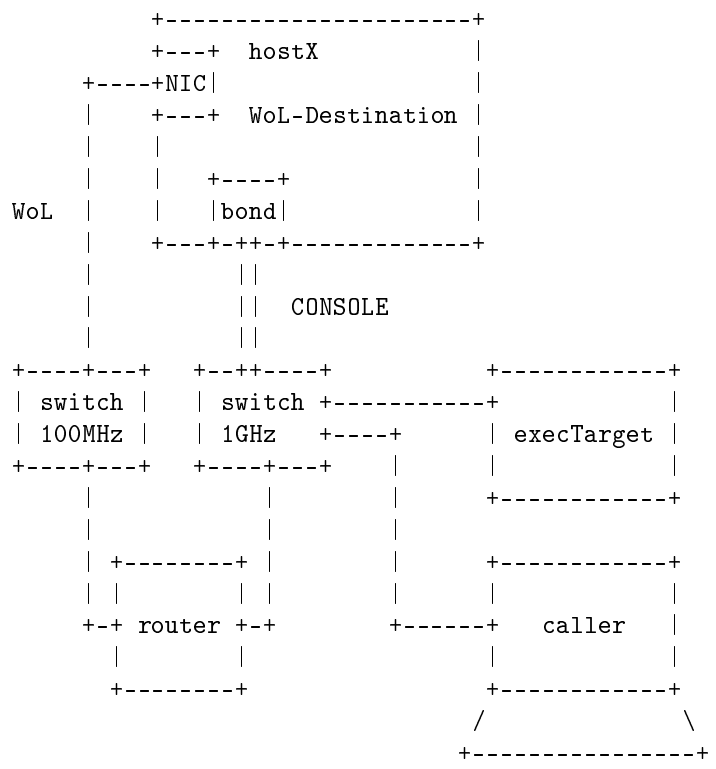

Figure 23.4: WoL configuration for a NIC behind a router

```
ctys \
 -t pm \
 -a create=wol,t:<hostX>,\
    broadcast:192.168.3.255 \
 -Z KSU \
```

```
 <execTarget>'(-Z KSU)'

ctys \
 -t pm \
 -a cancel=wol,if:eth2,self,stack,\
    poweroff:0 \
 -Z KSU \
 <hostX>'(-Z KSU)'
```

The reason for this scenario is the restriction of the NIC as beeing a recipient for WoL packets. This is defined so by the Motherboard/BIOS. But using a Gigabit port for multiple of those machines was not welcome. Thus the good-old 100MHz device, which is actually a c1538m, was used for WoL purposes only. It is used for a printerserver anyway, and had enough unused ports.

The router is a OpenBSD custom-made router, offering pass-through of directed-broadcasts, thus anything work fine.

23.1.5 Wake-On-Lan - Complete Reboot-Cycles

CLI

This halts the machine with preparation of WoL.

In this case sudo is used for raising permissions on the target machine where the sudoers is configured with "requiretty", thus the "-z 2" parameter is required.

```
ctys \
 -t pm \
 -a cancel=self,force,wol,poweroff:0,timeout:0 \
 -z 2 \
 wol1@lab01'(-Z sudo)'
```

The following call switches the machine "t:192.168.1.72" on and opens a CLI console within the callers window. Therefore the option "-z 2" is required twice and "-b 0"

once. This is due to the requirement of a pty for the reasons:

- local
  The remote call requires a pty for the sudo call, which has to be set locally when preparing the ssh-call.
- remote
  The remote call requires a pty for chained login to the target machine, once this is up and operable. The CLI of ssh requires in the test environment a pty.

The CLI requires in each case a synchronous forground mode by "-b 0".

```
ctys \
 -t pm \
 -a create=wol,t:192.168.1.72,sshping:wol1,\
    console:cli \
 -z 2 \
 -b 0 \
 lab00'(-Z sudo -z 2)'
```

XTERM

This halts the machine with preparation of WoL.

In this case sudo is used for raising permissions on the target machine where the sudoers is configured with "requiretty", thus the "-z 2" parameter is required.

```
ctys \
 -t pm \
 -a cancel=self,force,wol,poweroff:0,timeout:0 \
 -z 2 \
 wol1@lab01'(-Z sudo)'
```

The following call switches the machine "t:192.168.1.72" on and opens a "gnome-terminal" window with a "bash". Therefore the option "-z 2" is required for the execution of "ethtool" by sudoers, which is configured with "requiretty".

```
ctys \
 -t pm \
 -a create=wol,t:192.168.1.72,sshping:wol1,\
    console:xterm \
 -z 2  \
 lab00'(-Z sudo)'
```

GTERM

This halts the machine with preparation of WoL.

In this case sudo is used for raising permissions on the target machine where the sudoers is configured with "requiretty", thus the "-z 2" parameter is required.

```
ctys
 -t pm \
 -a cancel=self,force,wol,poweroff:0,timeout:0 \
 -z 2 \
 wol1@lab01'(-Z sudo)'
```

The following call switches the machine "t:192.168.1.72" on and opens a "gnome-terminal" window with a "bash". Therefore the option "-z 2" is required for the execution of "ethtool" by sudoers, which is configured with "requiretty".

```
ctys \
 -t pm \
 -a create=wol,t:192.168.1.72,sshping:wol1,\
    console:gterm \
 -z 2  \
 lab00'(-Z sudo)'
```

EMACS

This halts the machine with preparation of WoL.

In this case sudo is used for raising permissions on the target machine where the sudoers is configured with "requiretty", thus the "-z 2" parameter is required.

```
ctys \
 -t pm \
 -a cancel=self,force,wol,poweroff:0,timeout:0 \
 -z 2 \
 wol1@lab01'(-Z sudo)'
```

The following call switches the machine "t:192.168.1.72" on and opens a "gnome-terminal" window with a "bash". Therefore the option "-z 2" is required for the execution of "ethtool" by sudoers, which is configured with "requiretty".

```
ctys \
 -t pm \
 -a create=wol,t:192.168.1.72,sshping:wol1,\
    console:emacs \
 -z 2  \
 lab00'(-Z sudo)'
```

If Emacs does no start with a shell check your emacs version, your environment and your configuration first. Try the X11 plugin with "CONSOLE:EMACS" next.

VNC

This halts the machine with preparation of WoL.

In this case sudo is used for raising permissions on the target machine where the sudoers is configured with "requiretty", thus the "-z 2" parameter is required.

```
ctys \
 -t pm \
 -a cancel=self,force,wol,poweroff:0,timeout:0 \
 -z 2 \
 wol1@lab01'(-Z sudo)'
```

The following call switches the machine "t:192.168.1.72" on and opens a "gnome-terminal" window with a "bash". Therefore the option "-z 2" is required for the execution of "ethtool" by sudoers, which is configured with "requiretty".

```
ctys \
 -t pm \
 -a create=wol,t:192.168.1.72,sshping:wol1,console:vnc \
 -z 2  \
 lab00'(-Z sudo)'
```

Chapter 24

Common Session Options

24.1 Opening multiple ctyss

24.1.1 Multiple Calls

This creates 3 vncviewer desktops to 3 different hosts and one VMware-VM session, which will be done with one call when provided within one call or script:

```
ctys -a create=l:CONSOLE -g :1 host01
ctys -a create=l:CONSOLE -g :2 host02
ctys -a create=l:CONSOLE -g :3 host03
ctys -t vmw -a CREATE=f:'vmware/openbsd-001.vmx' \
     -g '600x400+2660+100' host01
```

24.1.2 One call

This call combines multiple session into one call, it opens 4 sessions on two hosts in seperate VNC desktops.

```
ctys  \
 --                                                  \
 host02'(-a create=l:F1,REUSE -g 400x400:3)'         \
 host02'(-a create=l:F2,REUSE -g 400x400+500+500:3)'\
 host01'(-a create=l:F3,REUSE -g 400x400:4)'         \
 host01'(-a create=l:F4,REUSE -g 400x400+500+500:4)'
```

This combines different sizes of client windows.

```
ctys \
 -b on                                                \
 --                                                   \
```

```
 host02'(-a create=l:chk1,REUSE -g 400x400+10+10:4)'  \
 host02'(-a create=l:chk2,REUSE -g 400x340+10+440:4)' \
 host02'(-a create=l:chk3,REUSE -g 400x400+420+10:4)' \
 host02'(-a create=l:chk4,REUSE -g 440x940+830+10:4)' \
 host02'(-a create=l:chk5,REUSE -g 800x130+10+820:4)' \
 host02'(-a create=l:chk6,REUSE -g 300x150+440+440:4)'\
 host02'(-a create=l:chk7,REUSE -g 300x150+490+630:4)'
```

24.2 Different resolution on client and server

This displays a local window of size "500x500" on screen 4. The resolution of the server (vncserver or Xclient) is "1800x1800":

```
ctys \
 -a create=l:CONSOLE \
 -g '500x500:4' \
 -r '1800x1800' \
 host01
```

24.3 Dynamic move of sessions window

For moving a vncviewer window circular across all screens and additionally within the screen, just the following lines are required.

```
#
#cycle screens
#
local _scr=; #screens
local _pos=; #offsets within that screen
local _POSLST="+600+100 +700+150 +800+200";
local _POSLST="$_POSLST +700+300 +600+200";
#
for (( _scr=0; _scr<7; _scr++));do
  #cycle within screen
  for pos in $_POSLST;do
    ctys \
      -a create=l:TST01,REUSE \
      -g "500x500${_pos}:$scr" \
      host01
    sleep 3
  done
```

```
done
```

The first call simply CREATE a ctys-session by starting vncserver, whereas the following calls just attach a vncviewer-client to the already running ctys-session. If the ctys-session already exists, no CREATE of ctys-session will be done.

This usage of REUSE here is utilized for the specific default behaviour of RealVNC(and tightVNC), where by default no desktop sharing is allowed. The standard behaviour is therefore after authorization to vncserver, to kill all (so one) previously started viewer-instances. When sharing is activated, REUSE just creates the requested new client, but does not touch the previous clients.

Thus for performance reasons here the call of REUSE is preferred, instead of using RECONNECT key, which first explicitly "kills" all previously locally started vncviewer instances on the current host of vncviewer execution for the for targeted ctys-session, which is the vncserver to be attached to.

The following does the same, but opens an additional VMware-Server session, which will be placed on a the screen next to VNC. Here it is performed by two seperate calls.

```
#
#cycle screens

#
local _scr=; #screens
local _pos=; #offsets within that screen
local _POSLST="+600+100 +700+150 +800+200";
local _POSLST="$_POSLST +700+300 +600+200";
#
for (( _scr=3; _scr<7; _scr++));do
  #cycle within screen
  for pos in $_POSLST ;do

    ctys -a create=l:TST01,REUSE            \
         -g "500x500${_pos}:${scr}"         \
         host01
```

```
    ctys -t VMW                                \
         -a create=l:TST01-VMW,RECONNECT       \
         -g "500x500${_pos}:$((scr+1))"        \
         host01

    sleep 3
  done
done
```

The following does the same, but performs just one call for both.

```
#
#cycle screens
#
local _scr=; #screens
local _pos=; #offsets within that screen
local _POSLST="+600+100 +700+150 +800+200";
local _POSLST="$_POSLST +700+300 +600+200";
#
for (( _scr=3; _scr<7; _scr++));do
  #cycle within screen
  for pos in $_POSLST;do

    ctys                                       \
       host01"(                                \
          -a create=l:TST01,REUSE              \
          -g 500x500${_pos}:${scr}             \
       )"                                      \
       host01"(                                \
          -t VMW                               \
          -a create=l:TST01-VMW,RECONNECT      \
          -g "500x500${_pos}:$((scr+1))"       \
       )"

    sleep 3
  done
done
```

The following does the same and adds XEN. Which has to be executed on a different host than VMware is.

```
#
#cycle screens
#
```

```
local _scr=; #screens
local _pos=; #offsets within that screen
local _POSLST="+600+100 +700+150 +800+200";
local _POSLST="$_POSLST +700+300 +600+200";
#
for (( _scr=3; _scr<7; _scr++));do
  #cycle within screen
  for pos in $_POSLST;do

    ctys                                        \
        host01"(                                \
            -t VNC                              \
            -a create=l:TST01,REUSE             \
            -g 500x500${_pos}:${scr}            \
        )"                                      \
        host01"(                                \
            -t VMW                              \
            -a create=l:TST01-VMW,RECONNECT     \
            -g "500x500${_pos}:$((scr+1))"      \
        )"                                      \
        host02"(                                \
            -t XEN                              \
            -a create=l:TST01-XEN,RECONNECT     \
            -g "500x500${_pos}:$((scr+2))"      \
        )"

    sleep 10 #give some more time
  done
done
```

Now adding multiple desktop support.

```
#
#cycle screens
#
local _scr=; #screens
local _pos=; #offsets within that screen
local _POSLST="+600+100 +700+150 +800+200";
local _POSLST="$_POSLST +700+300 +600+200";
#
for (( _scr=3; _scr<7; _scr++));do
  for desk in rd prod 3;do  #rd=1, prod=2, 3=admin
  #cycle within screen
```

```
    for pos in $_POSLST;do

      ctys                                          \
        host01"(                                    \
          -t VNC                                    \
          -a create=l:TST01,REUSE                   \
          -g 500x500${_pos}:${scr}                  \
          -D ${desk}                                \
        )"                                          \
        host02"(                                    \
          -t VMW                                    \
          -a create=l:TST01-VMW,RECONNECT           \
          -g "500x500${_pos}:$((scr+1))"            \
          -D ${desk}                                \
        )"                                          \
        host03"(                                    \
          -t XEN                                    \
          -a create=l:TST01-XEN,RECONNECT           \
          -g "500x500${_pos}:$((scr+2))"            \
          -D ${desk}                                \
        )"

    done

    sleep 10 #give some more time
  done
done
```

24.4 Spanning Multiple Physical Screens

Currently it seems though, that in case of RealVNC 4.1.2 on CentOS-5.0 the following behaviour is given:

- A desktop with size bigger than a physical(better to say X11-configured) screen has to be configured when starting the vncserver. So e.g. the geometry parameter for the vncserver call could be "2560x1024" on a adjecent pair of "1280x1024" screens.
- When opening the vncviewer the desktop will be sized as given by the vncserver, but the displayed window is restricted by the current screen dimension.

As it seems to be, the oversized window is restricted to the size of the screen of the monitor in the middle of the cal-

culated area of required size and offset. This is the device on which the resulting screen will be displayed. The maximum size given as an option to vncviewer will not change this behaviour.

This seems to be a bug of vncviewer, but anyhow, the horizontal scrollbar shows the correct proportions and the windows could be expanded manually to it's full size, spanning multiple screens.

This behaviour is a major drawback for automatic and final configuration of desktop layouts, requiring additional manual interaction. But it does only effect, when sizing a window bigger than a screen.

So with the following call a window could be created, which is spanning 4 adjecent screens, but has to expanded to it's final size manually:

```
ctys \
 -a create=l:BIG13 \
 -g '5120x1024:3' \
 host01
```

For debugging of the remote actions the following could be called:

```
ctys \
 -a create=l:BIG13 \
 -g '5120x1024:3' \
 -- \
 '(-d 6)' host01
```

Which sets the debugging level on the remote host "host01" to "6".

This spans now 2 screens:

```
ctys \
 -a create=l:CONSOLE \
 -g "2560+0:1" \
 host01
```

This resets to one screen:

```
ctys \
```

```
-a create=l:CONSOLE \
-g ":1" \
host01
```

24.5 CREATE with tree-search for unique IDs

One of the real smart features of ctys is it's ability to use any of it's unique IDs for automatic search for related vmx-file within a defined subtree, which is by default HOME.

Any directory prefix could be provided.

The applicable IDs are

- the vmx-filename with any partial relative path-prefix
- the UUID
- the LABEL, a.k.a displayName of name/DomainName.

The following calls are possible:

- This starts a VMware session with the first matched UUID within subtree "vmware/dir2" relative to HOME on host01.

  ```
  ctys \
   -t vmw \
   -a create=uuid:0101010...01,\
      basepath:vmware/dir2 \
   host01
  ```

- This starts a VMware session with the first matched LABEL within subtree "vmware/dir2" relative to HOME on host01.

  ```
  ctys \
   -t vmw \
   -a create=label:MatchMe,basepath:vmware/dir2 \
   host01
  ```

- This starts a VMware session with the first matched vmx-file within subtree "vmware/dir2" relative to HOME on host01.

  ```
  ctys \
   -t vmw \
   -a create=filename:dir2/OpenBSD-01/OpenBSD-01.vmx,\
      basepath:vmware \
  ```

```
 host01
```

- This starts a VMware session with the vmx-file on host01.

```
ctys \
 -t vmw \
 -a create=pname:\$HOME/dir2/OpenBSD-01/OpenBSD-01.vmx \
 host01
```

24.6 Some session related calls

24.6.1 ENUMERATE

Enumerate the current available sessions of type VMW, where UUIDs are displayed additionaly to labels and vmx-files. The scan for vmx-files begins relative to the callers HOME directory within the subdirectories: "vmware/dir2" and "vmware/dir3" on host01.

```
ctys \
 -t vmw \
 -a enumerate=UUID,vmware/dir2\%vmware/dir3 \
 host1
```

24.6.2 LIST

List all current active sessions, where all attributes are visible, the fullpathname for vmx-files is displayed. Initially all type-plugins are loaded and listed due to load state. All CLIENT and SERVER processes located on the host01 are displayed. Warnings are suppressed.

```
ctys \
 -a list=all,fullpath,both \
 -W \
 -T \
 all host01
```

HINT:Loading of all present plugins could exhaust shell resources.

24.6.3 SHOW

This shows the current dynamic state of the remote hosts, therefore basic system information for OS, MACHINE, RAM,

processes(by top), and current ALARMS of lm_sensors if installed.

```
ctys -a show host0{1,2}
```

24.6.4 INFO

This displays information related to static data of selected hosts, which contains installed OS, CPU-info, RAM-info, VNC-info, and wmctrl-info.

For practical purposes the CPU-Flags: VT-x, AMD-V, and PAE are displayed.

```
ctys -a info host0{1,2}
```

24.7 Some ctys related calls

24.7.1 Display Version and available plugin

The following call enumerates all actually loaded plugins as set by default:

```
ctys -v
```

The following call enumerates all actually available plugins and their versions.

```
ctys -v -T all
```

Chapter 25

Custom CLI

25.1 Groups

One of the most valuable features for the setup of custom desktops is the groups feature. Any setup of a X11 desktop including multiple desktops could be prepared and executed.

The following example illustrates the setup of a workspace with some basic remote desktops for their management. In this example the user root is used without encapsulation of the calls by sudo or ksu.

The interactive call for the group file named "admin" is:

```
ctys admin
```

Where the group file has the following content.

```
#
#This groups contains all machines in the
#management group.
#
root@machine1'(-t vnc -a create=reuse,l:MACHINE1 \
               -g 1268x872:A20 -D admin -b 1,2)'
root@machine2'(-t vnc -a create=reuse,l:MACHINE2 \
               -g :A30 -D admin -b 1,2)'
root@machine3'(-t vnc -a create=reuse,l:MACHINE3 \
               -g :A00 -D admin -b 1,2)'
root@machine4'(-t vnc -a create=reuse,l:MACHINE4 \
```

```
                -g :A01 -D admin -b 1,2)'
root@machine5'(-t vnc -a create=reuse,l:MACHINE5
                -g :A21 -D admin -b 1,2)'
```

This example shows a number of specifics to be considered.

First of all, the ssumption is made, that each root account has to be authorized by an interactive password request. Therefore the "-b 1,2" option is required. This forces a background but sequential execution, which causes a non-intermixed and sequential password request, but once authorized, the process is detached from the console. Thus multiple interactive password requests for daemons could be provided.

The second point to be recognized is the complete support of the type and action information within the context options . This allows the intermixed usage of several session types within one call.

The next point is the usage of the "-D admin" option for the display of all admin tasks on the "admin" workspace, where the alias "admin" is a custom definition by the user.

The same resulting call could be provided by the following variant, which is more flexible, but requires therefore some additional call parameters. The interactive call for the group file named "admin" is now:

```
ctys -l root -b 1,2 admin
```

The "-l root" option sets the target user to be used for all resulting targets from the group "admin". The "-b 1,2" sets the value to be used for all subsequent internal subcalls.

The group file has now the following content.

```
#
#This groups contains all machines in the
#management group.
#
machine1'(-t vnc -a create=reuse,l:MACHINE1 \
          -g 1268x872:A20 -D admin)'
```

```
machine2'(-t vnc -a create=reuse,l:MACHINE2 \
          -g :A30 -D admin)'
machine3'(-t vnc -a create=reuse,l:MACHINE3 \
          -g :A00 -D admin)'
machine4'(-t vnc -a create=reuse,l:MACHINE4 \
          -g :A01 -D admin)'
machine5'(-t vnc -a create=reuse,l:MACHINE5
          -g :A21 -D admin)'
```

25.2 Macros

25.3 Tables

25.3.1 Common Tables for ENUMERATE and LIST

This example shows a table definition for LIST and ENUMERATE the CPORT - ClientPort - of a plugin. This is usually the port for access by vncviewer.

```
"tab_gen:3_Label_10%%macro:F_STYPE%%9_cport_5%%\
  1_PM_15%%6_MAC_18%%7_TCP_15"
```

The defintion is used literally within LIST action as:

```
ctys \
 -a list=tab_gen:3_Label_10%%macro:F_STYPE%%\
    9_cport_5%%1_PM_15%%6_MAC_18%%7_TCP_15\
 lab00 lab01
```

The defintion is used literally within ENUMERATE action as:

```
ctys \
 -a enumerate=tab_gen:3_Label_10%%macro:F_STYPE%%\
    9_cport_5%%1_PM_15%%6_MAC_18%%7_TCP_15\
 lab00 lab01
```

The same definition could be used to define a macro for a table.

```
TAB_CPORT=tab_gen:3_Label_10%%macro:F_STYPE%%\
  9_cport_5%%1_PM_15%%6_MAC_18%%7_TCP_15
```

The macro and used within LIST action as:

```
ctys -a list=macro:TAB_CPORT lab00 lab01
```

The result of the LIST action example is:

```
    Label   |stype |cport|PM        |MAC               |TCP
    --------+------+-----+----------+------------------+------------
    testx11 |CLI   |     |lab00.soho|                  |
    testx11 |CLI   |     |lab00.soho|                  |
    LAB00   |VNC   |5901 |lab00.soho|                  |
    tst     |VNC   |5902 |lab00.soho|                  |
    Domain-0|XEN   |     |lab00.soho|                  |
    tst101  |XEN   |5928 |lab00.soho|00:50:56:13:11:41 |
    lab00   |PM    |     |lab00.soho|00:0E:0C:35:F8:48 |192.168.1.71
    Domain-0|XEN   |     |lab01.soho|                  |
    lab01   |PM    |     |lab01.soho|00:0E:0C:C3:CD:12 |192.168.1.72
    tst000  |VNC   |     |ws2.soho  |                  |
    tst001  |VNC   |     |ws2.soho  |                  |
```

The macro used within ENUMERATE action as:

```
ctys -a enumerate=macro:TAB_CPORT lab00 lab01
```

The result of the ENUMERATE action example is:

```
Label     |stype|cport|PM        |MAC               |TCP
----------+-----+-----+----------+------------------+------------
sparc-1   |QEMU |     |lab00.soho|00:50:56:13:11:49|QEMU
sparc-1   |QEMU |     |lab00.soho|00:50:56:13:11:49|QEMU
coldfire-t|QEMU |     |lab00.soho|00:50:56:13:11:49|QEMU
arm-test  |QEMU |     |lab00.soho|00:50:56:13:11:49|QEMU
linux     |QEMU |     |lab00.soho|00:50:56:13:11:49|QEMU
small     |QEMU |     |lab00.soho|00:50:56:13:11:49|QEMU
qemu-tst01|QEMU |     |lab00.soho|00:50:56:13:11:52|QEMU
arm-test  |QEMU |     |lab00.soho|00:50:56:13:11:49|QEMU
small     |QEMU |     |lab00.soho|00:50:56:13:11:49|QEMU
sparc-1   |QEMU |     |lab00.soho|00:50:56:13:11:49|QEMU
sparc-1   |QEMU |     |lab00.soho|00:50:56:13:11:49|QEMU
linux     |QEMU |     |lab00.soho|00:50:56:13:11:49|QEMU
coldfire-t|QEMU |     |lab00.soho|00:50:56:13:11:49|QEMU
linux001  |VMW  |     |lab00.soho|00:50:56:15:11:01|192.168.1.150
linux002  |VMW  |5977 |lab00.soho|00:50:56:15:11:02|192.168.1.151
tst100    |XEN  |     |lab00.soho|00:50:56:13:11:40|
tst100    |XEN  |     |lab00.soho|00:50:56:13:11:40|
tst100    |XEN  |     |lab00.soho|00:50:56:13:11:40|
tst100    |XEN  |     |lab00.soho|00:50:56:13:11:40|
tst101    |XEN  |     |lab00.soho|00:50:56:13:11:41|
tst101    |XEN  |     |lab00.soho|00:50:56:13:11:41|
lab00     |PM   |     |lab00.soho|00:0E:0C:35:F8:48|192.168.1.71
```

25.3.2 RAW Tables by MACHINE

25.3.3 Combined MACROS and Tables

The following examples shows stored table options for LIST and ENUMERATE action within macros.

Several of the predefined tables are generic, thus containing fields available from the display methods "ctys -a ENUMERATE=...", "ctys -a LIST=...", and "ctys-vhost -o tab_gen:...", thus these could be used literally in all of them, just the wrapping call convention varies a little. Thus the macro definitions for generic tables are used unaltered, only adaptive call convention superpositioning macros are defined.

The pure text replacement by macros allows for additional suboptions, when these are seamless concatenated by usage of an intermediate field seperator. Thus the content of a macro could be expanded without altering it's definition, even though in this draft version no specific macroseperator is defined.

```
#!/bin/bash #4syncolors
#########################################################
#
#PROJECT:     Unified Sessions Manager
#AUTHOR:      Arno-Can Uestuensoez -
#               acue@UnifiedSessionsManager.org
#MAINTAINER:  Arno-Can Uestuensoez -
#               acue_sf1@sourceforge.net
#SHORT:       ctys
#CALLFULLNAME:Commutate To Your Session
#LICENCE:     GPL3
#VERSION:     01_06_001a10
#
#########################################################
#
#Copyright(C) 2008 Arno-Can Uestuensoez
#  (UnifiedSessionsManager.org)
#
#This program is free software: you can redistribute
#it and/or modify it under the terms of the GNU General
#Public License as published by the Free Software
#Foundation, either version 3 of the License, or (at
```

```
#your option) any later version.
#
#This program is distributed in the hope that it will
#be useful, but WITHOUT ANY WARRANTY; without even the
#implied warranty of MERCHANTABILITY or FITNESS FOR A
#PARTICULAR PURPOSE.  See the GNU General Public
#License for more details.
#
#You should have received a copy of the GNU General
#Public License along with this program.  If not,
#see <http://www.gnu.org/licenses/>.
#
############################################################


############################################################
#
#Atoms with appropriate sizes.
#
#
#ContainingMachine(1);
#SessionType(2);
#Label(3);
#ID(4);
#UUID(5);
#MAC(6);
#TCP(7);
#DISPLAY(8);
#ClientAccessPort(9);
#VNCbasePort(10);
#
F_PM        = 1_PM_15
F_STYPE     = 2_stype_10
F_LABEL     = 3_label_10
F_ID        = 4_ID_25_L
F_UUID      = 5_UUID_32
F_MAC       = 6_MAC_18
F_TCP       = 7_TCP_15
F_DISP      = 8_DISP_4
F_CPORT     = 9_cport_5_L
F_SPORT     = 10_sport_5_L

############################################################
```

```
#
#Specific additional MACROS for LIST
#PID(11);
#UID(12);
#GUID(13);
#C/S-Type(14)
#
F_PID       = 11_pid_5
F_UID       = 12_uid_8
F_GUID      = 13_gid_8
F_CST       = 14_cst_1


#########################################################
#
#Specific additional MACROS for ENUMERATE
#
#VNCbaseport(11);
#Distro(12);
#Distrorel(13);
#OS(14);
#OS(15);
#VersNo(16);
#SerialNo(17);
#Category(18)
#VMstate(19)
#hyperrel(20)
#StackCap(21)
#StackReq(22)
#HWcap(23)
#HWreq(24)
#execloc(25)
#reloccap(26)
#SSH(27)
#rsrv(28)
#rsrv(29)
#rsrv(30)
#rsrv(31)
#rsrv(32)
#rsrv(33)
#CTYSrel(34)
#netmask(35)
#Gateway(36)
```

```
#Relay(37)
#Arch(38)
#Platform(39)
#VRAM(40)
#VCPU(41)
#ContextStg(42)
#UserStrg(43)
#
F_VNCBASE   = 11_vncbase_7
F_DIST      = 12_distro_12
F_DISTREL   = 13_distrorel_15
F_OS        = 14_os_10
F_OSREL     = 15_osrel_10
F_VERNO     = 16_verno_9
F_SERNO     = 17_serno_14
F_CATEGORY  = 18_category_8
F_VMSTATE   = 19_VMstate_9
F_HYPERREL  = 20_hyperrel_15
F_STACKCAP  = 21_StackCap_15_B
F_STACKREQ  = 22_StackReq_15_B
F_HWCAP     = 23_HWcap_60_B
F_HWREQ     = 24_HWreq_25_B
F_EXECLOC   = 25_execloc_15
F_RELOCCAP  = 26_reloccap_8
F_SSHPORT   = 27_SSH_5
F_RSRV6     = 28_r_1
F_RSRV7     = 29_r_1
F_RSRV8     = 30_r_1
F_RSRV9     = 31_r_1
F_RSRV10    = 32_r_1
F_IFNAME    = 33_if_7
F_CTYSREL   = 34_CTYSrel_10
F_NETMASK   = 35_netmask_15
F_GATEWAY   = 36_Gateway_15
F_RELAY     = 37_Relay_15
F_ARCH      = 38_Arch_6
F_PLATFORM  = 39_Platform_10
F_VRAM      = 40_VRAM_5
F_VCPU      = 41_VCPU_4
F_CSTRG     = 42_ContextStg_20_B
F_USTRG     = 43_UserStrg_20_B
```

```
#########################################################
#
#DEFAULT for ctys-vhost, when no "-o" option is
#selected. Change this carefully, otherwise ctys-vhost
#might come into trouble.
#
TAB_CTYS_VHOST_DEFAULT=tab_gen:macro:F_LABEL%%\
macro:F_STYPE%%macro:F_DIST%%macro:F_DISTREL%%\
macro:F_OS%%macro:F_OSREL%%macro:F_PM%%macro:F_TCP

listdefault=-a list=macro:TAB_CTYS_VHOST_DEFAULT
ldefault=-a list=macro:TAB_CTYS_VHOST_DEFAULT

enumdefault=-a enumerate=macro:TAB_CTYS_VHOST_DEFAULT
edefault=-a enumerate=macro:TAB_CTYS_VHOST_DEFAULT

vhostdefault=-o macro:TAB_CTYS_VHOST_DEFAULT
vdefault=-o macro:TAB_CTYS_VHOST_DEFAULT


#########################################################
#
#Basic hypervisor state
#
TAB_HYPER=tab_gen:macro:F_LABEL%%macro:F_STYPE%%\
macro:F_VMSTATE%%macro:F_OS%%macro:F_OSREL%%\
macro:F_ARCH%%macro:F_VCPU%%macro:F_VRAM

enumhyper=-a enumerate=macro:TAB_HYPER
ehyper=-a enumerate=macro:TAB_HYPER

vhosthyper=-o macro:TAB_HYPER
vhyper=-o macro:TAB_HYPER


#########################################################
#
#Basic stack state
#
TAB_STACKSTAT=tab_gen:macro:F_LABEL%%macro:F_STYPE%%\
macro:F_VMSTATE%%macro:F_OS%%macro:F_OSREL%%\
macro:F_STACKCAP%%macro:F_STACKREQ
```

```
enumstack=-a enumerate=macro:TAB_STACKSTAT
estack=-a enumerate=macro:TAB_STACKSTAT

vhoststack=-o macro:TAB_STACKSTAT
vstack=-o macro:TAB_STACKSTAT


#########################################################
#
#connections with PID
#
# LABEL STYPE  DISP  CPORT  SPORT  PID PM  TCP
#
TAB_LST_CONNPID=tab_gen:macro:F_LABEL%%\
macro:F_STYPE%%macro:F_CST%%macro:F_DISP%%\
macro:F_CPORT%%macro:F_SPORT%%\
macro:F_PID%%macro:F_PM%%macro:F_TCP

connpid=macro:TAB_LST_CONNPID
listconnpid=-a list=macro:TAB_LST_CONNPID


#########################################################
#
#connections
#
# LABEL STYPE  DISP  CPORT  SPORT  PM  TCP
#
TAB_ENUMLST_CONNECT=tab_gen:macro:F_LABEL%%\
macro:F_STYPE%%macro:F_DISP%%macro:F_CPORT%%\
macro:F_SPORT%%macro:F_PM%%macro:F_TCP

conn=macro:TAB_ENUMLST_CONNECT
lconn=-a list=macro:TAB_ENUMLST_CONNECT
econn=-a enumerate=macro:TAB_ENUMLST_CONNECT
vconn=-o macro:TAB_ENUMLST_CONNECT


#########################################################
#
#interfaces
#
```

```
# LABEL STYPE  PM  TCP MAC
#
TAB_ENUMLIST_INTERFACES=tab_gen:macro:F_LABEL%%\
macro:F_STYPE%%macro:F_PM%%macro:F_TCP%%macro:F_MAC

interfaces=macro:TAB_ENUMLIST_INTERFACES
lif=-a list=macro:TAB_ENUMLIST_INTERFACES
eif=-a enumerate=macro:TAB_ENUMLIST_INTERFACES
vif=-o macro:TAB_ENUMLIST_INTERFACES


#########################################################
#
#conffiles
#
# LABEL STYPE  TCP  MAC  ID
#
TAB_ENUMLIST_CONF=tab_gen:macro:F_LABEL%%\
macro:F_STYPE%%macro:F_TCP%%macro:F_MAC%%4_ID_50_L

conf=macro:TAB_ENUMLIST_CONF
lconf=-a list=macro:TAB_ENUMLIST_CONF
econf=-a enumerate=macro:TAB_ENUMLIST_CONF
vconf=-o macro:TAB_ENUMLIST_CONF


#########################################################
#
#ids
#
# LABEL STYPE  TCP  MAC  UUID  ID
#
TAB_ENUMLIST_ID=tab_gen:macro:F_LABEL%%macro:F_STYPE%%\
macro:F_TCP%%macro:F_MAC%%macro:F_UUID%%macro:F_ID

id=macro:TAB_ENUMLIST_ID
lid=-a list=macro:TAB_ENUMLIST_ID
eid=-a enumerate=macro:TAB_ENUMLIST_ID
vid=-o macro:TAB_ENUMLIST_ID
```

Usage with VMSTATE

The predefined macros could be expanded by additional attributes, when these are resulting in an overall semantically correct definition after final expansion.

One example application is the usage of the provided standard macro "enumhyper", which defines a table containing the main hypervisor and GuestOS attributes. The standard of ctys for ENUMERATE call will be:

```
ctys -t vmw macro:enumhyper
```

The result of the ENUMERATE action is:

```
label     |stype |VMstate|os     |osrel|Arch|VCPU|VRAM
----------+------+-------+-------+-----+----+----+-----
tst117    |VMW   |ACTIVE |Linux  |2.6  |    |    |
tst115    |VMW   |ACTIVE |Solaris|10   |    |    |
tst116    |VMW   |ACTIVE |Linux  |2.6  |    |    |
tst112    |VMW   |ACTIVE |Linux  |2.6  |    |    |
tst003    |VMW   |ACTIVE |Linux  |2.6  |    |    |
tst005    |VMW   |ACTIVE |Linux  |2.6  |    |    |
tst103    |VMW   |ACTIVE |Linux  |2.6  |    |    |
tst106    |VMW   |ACTIVE |Linux  |2.6  |    |    |
tst111    |VMW   |ACTIVE |OpenBSD|4.2  |    |    |
tst120    |VMW   |ACTIVE |FreeBSD|6.1  |    |    |
tst128    |VMW   |ACTIVE |NetBSD |4.0  |    |    |
tst002    |VMW   |ACTIVE |Linux  |2.6  |    |    |
tst132    |VMW   |ACTIVE |Linux  |2.6  |    |    |
tst133    |VMW   |       |Linux  |2.6  |    |    |
tst155    |VMW   |       |OpenBSD|4.3  |    |    |
tst109    |VMW   |ACTIVE |OpenBSD|4.0  |    |    |
GRP02     |VMW   |       |other  |     |    |    |
GRP01-open|VMW   |       |other  |     |    |    |
GRP02     |VMW   |       |other  |     |    |    |
linux001  |VMW   |       |Linux  |2.6  |    |    |
linux002  |VMW   |       |Linux  |2.6  |    |    |
linux001  |VMW   |       |Linux  |2.6  |    |    |
```

This displays the default output of "MATCHVSTAT=ACTIVE%EMPTY". When the "MATCHVSTAT=ACTIVE" subset is required

only the following call-extension to the macro could be applied.

```
ctys -t vmw macro:enumhyper,matchvstat:active
```

The "-t vmw" option sets SESSIONTYPE to VMW, thus all other are ignored. Additionally the macro "macro:enumhyper" is expanded by ",matchvstat:active" in order to display onyl active sessions, where the field VMSTATE has the kiteral value "ACTIVE". It is important to recognize the comma "," as field seperator here.

The result of the ENUMERATE action is:

```
label |stype |VMstate  |os        |osrel |Arch  |VCPU|VRAM
------+------+---------+----------+------+------+----+-----
tst117|VMW   |ACTIVE   |Linux     |2.6   |      |    |
tst115|VMW   |ACTIVE   |Solaris   |10    |      |    |
tst116|VMW   |ACTIVE   |Linux     |2.6   |      |    |
tst112|VMW   |ACTIVE   |Linux     |2.6   |      |    |
tst003|VMW   |ACTIVE   |Linux     |2.6   |      |    |
tst005|VMW   |ACTIVE   |Linux     |2.6   |      |    |
tst103|VMW   |ACTIVE   |Linux     |2.6   |      |    |
tst106|VMW   |ACTIVE   |Linux     |2.6   |      |    |
tst111|VMW   |ACTIVE   |OpenBSD   |4.2   |      |    |
tst120|VMW   |ACTIVE   |FreeBSD   |6.1   |      |    |
tst128|VMW   |ACTIVE   |NetBSD    |4.0   |      |    |
tst002|VMW   |ACTIVE   |Linux     |2.6   |      |    |
tst132|VMW   |ACTIVE   |Linux     |2.6   |      |    |
tst109|VMW   |ACTIVE   |OpenBSD   |4.0   |      |    |
```

The previous example is now expanded to display additionally VMs with the state "TESTDUMMY", which are the test configuration files contained in the current version. These contain testpattern only, not foreseen to be used.

```
ctys -t vmw macro:enumhyper,matchvstat:active%testdummy
```

The result of the ENUMERATE action now contains additionally the available test-pattern:

```
label     |stype |VMstate   |os     |osrel |Arch|VCPU|VRAM
----------+------+---------+-------+------+----+----+-----
tst117    |VMW   |ACTIVE    |Linux  |2.6   |    |    |
tst115    |VMW   |ACTIVE    |Solaris|10    |    |    |
tst116    |VMW   |ACTIVE    |Linux  |2.6   |    |    |
tst112    |VMW   |ACTIVE    |Linux  |2.6   |    |    |
tst003    |VMW   |ACTIVE    |Linux  |2.6   |    |    |
tst005    |VMW   |ACTIVE    |Linux  |2.6   |    |    |
tst103    |VMW   |ACTIVE    |Linux  |2.6   |    |    |
tst106    |VMW   |ACTIVE    |Linux  |2.6   |    |    |
tst111    |VMW   |ACTIVE    |OpenBSD|4.2   |    |    |
tst120    |VMW   |ACTIVE    |FreeBSD|6.1   |    |    |
tst128    |VMW   |ACTIVE    |NetBSD |4.0   |    |    |
tst002    |VMW   |ACTIVE    |Linux  |2.6   |    |    |
tst132    |VMW   |ACTIVE    |Linux  |2.6   |    |    |
tst109    |VMW   |ACTIVE    |OpenBSD|4.0   |    |    |
tst01vmx  |VMW   |TESTDUMMY|1      |1     |1   |1   |2
tst05vmx  |VMW   |TESTDUMMY|1      |1     |1   |1   |2
tst11vmx  |VMW   |TESTDUMMY|1      |1     |1   |1   |2
tst11vmx  |VMW   |TESTDUMMY|1      |1     |1   |1   |2
tst11vmx  |VMW   |TESTDUMMY|1      |1     |1   |1   |2
tst11vmx  |VMW   |TESTDUMMY|1      |1     |1   |1   |2
tst11vmx  |VMW   |TESTDUMMY|1      |1     |1   |1   |2
tst11vmx  |VMW   |TESTDUMMY|1      |1     |1   |1   |2
tst11vmx  |VMW   |TESTDUMMY|1      |1     |1   |1   |2
```

The same where the rable is sorted by the 4-th column of the final result.

```
ctys -t vmw macro:enumhyper,matchvstat:active%testdummy,sort:4
```

The result of the ENUMERATE action now is additionally sorted by the "os" filed.

label	stype	VMstate	os	osrel	Arch	VCPU	VRAM
tst01vmx	VMW	TESTDUMMY	1	1	1	1	2
tst05vmx	VMW	TESTDUMMY	1	1	1	1	2
tst11vmx	VMW	TESTDUMMY	1	1	1	1	2
tst11vmx	VMW	TESTDUMMY	1	1	1	1	2
tst11vmx	VMW	TESTDUMMY	1	1	1	1	2
tst11vmx	VMW	TESTDUMMY	1	1	1	1	2
tst11vmx	VMW	TESTDUMMY	1	1	1	1	2
tst11vmx	VMW	TESTDUMMY	1	1	1	1	2
tst11vmx	VMW	TESTDUMMY	1	1	1	1	2
tst120	VMW	ACTIVE	FreeBSD	6.1			
tst002	VMW	ACTIVE	Linux	2.6			
tst003	VMW	ACTIVE	Linux	2.6			
tst005	VMW	ACTIVE	Linux	2.6			
tst103	VMW	ACTIVE	Linux	2.6			
tst106	VMW	ACTIVE	Linux	2.6			
tst112	VMW	ACTIVE	Linux	2.6			
tst116	VMW	ACTIVE	Linux	2.6			
tst117	VMW	ACTIVE	Linux	2.6			
tst132	VMW	ACTIVE	Linux	2.6			
tst128	VMW	ACTIVE	NetBSD	4.0			
tst109	VMW	ACTIVE	OpenBSD	4.0			
tst111	VMW	ACTIVE	OpenBSD	4.2			
tst115	VMW	ACTIVE	Solaris	10			

Chapter 26

Pre-Configured Desktops

26.1 Admin RAID-Info

The following example shows a preconfiguration of an example Desktop, where the health of the basic resources is inspected in detail. Therefore the main RAID systems are inspected and monitored by their proprietary tools, and the UPS supplying to the systems is displayed by the OpenSource "netups". Each remote host is displayed by VNC on a seperate screen, but could be moved and resized throughout the whole array as required.

The access to all of this tools is due to general philosophy permitted to local access only, thus requires an SSH session provided here by UnifiedSessionsManager. Direct remote access is either disabled by configuring UNIX-Domain sockets or by blocking the communications.

This example contains sessions of the plugin VNC only, thus representing HOSTS sessions to physical machines runnning their native OS(and/or Dom0 for XEN). No startup of a VM by an involved hypervisor is required.

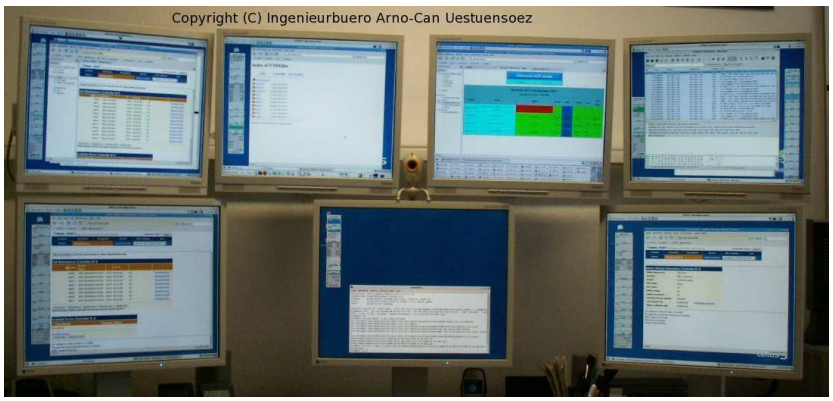

Figure 26.1: Administration of RAID and Power-Supply

The display array is configured as Xinerama mode for array-like addressing as described in Section 5.2.1 'Logical Layout-1b' on page 57 . Which has the following namebinding for the physical array-parts of the logical screen.

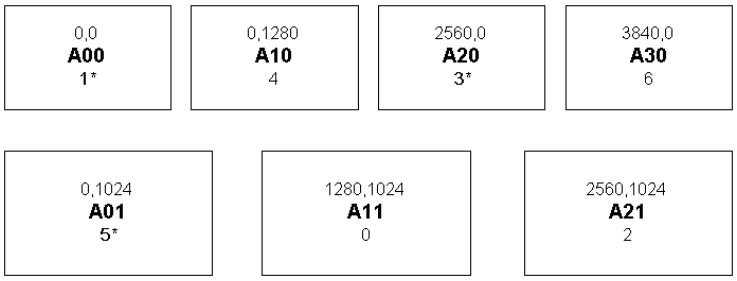

Figure 26.2: Logical Multi-Screen X11-Array-Style

Thus the screens with the logical address A00, A10, and A02 display the 3ware raid tool 3DM2 and "gkrellm".

The screen A20 displays the management station with "netups" for the UPS array.

The screen A30 displays the out-ot-band protocol analyser based on ethereal/wire-shark and gkrellm.

The screen A10 displays the available UNIX-Distros in the html-view.

The screen A11 is reserved as console for interaction.

The reason of describing such a scenario is the each-time effort required, when starting this desktop manually. Things become even worster, when the "moderate" count of sessions displayed on the "taskmanager" at the bottom of screen A20 is considered. Thus a manual startup of a multi-desktop/workspace environment could easily take 10minutes and more - each time.

The customizing with ad-hoc-written-scripts for each case is viable for an overseable number of cases only, it becomes cumbersome quickly, when expanding the pre-defined scenarios and number of involved sessions just a little.

Therefore ths UnifiedSessionsManager was designed and implemented, in order to ease the definition of masses of scenarios with involved bulks of sessions. To shorten it up, the actual call to startup the above destop is:

ctys admin

That's it.

The call consists of the mandatory base call "ctys" of course, and an additional pre-configure group called "admin". The goup contains the following entries:

```
#
#This groups contains all machines in the management group.
#
root@host1'( -t vnc -a create=reuse,l:HOST1 \
             -g 1268x872:A20 \
             -D admin -b 1,2)'

root@host2'( -t vnc -a create=reuse,l:HOST2 \
             -g 1268x994:A30 \
             -D admin -b 1,2)'

root@host3'( -t vnc -a create=reuse,l:HOST3 \
             -g 1268x994:A00 \
             -D admin -b 1,2)'
```

```
root@host4'( -t vnc -a create=reuse,l:HOST4 \
             -g 1268x958:A10 \
             -D admin -b 1,2)'

root@host5'( -t vnc -a create=reuse,l:HOST5 \
             -g 1268x994:A01 \
             -D admin -b 1,2)'

root@host6'( -t vnc -a create=reuse,l:HOST6 \
             -g 1268x994:A21 \
             -D admin -b 1,2)'
```

where each entry itself is a complete context specific configuration of a call.

The most remarkable option is here the "-g" option, which defines the display size and position. The specifics for VNC to consider here is the missing "-r" option, which defines the actual server-resolution, whereas the g-option just defines the size of the client window. The "-r" option is actually supported for VNC only dynamically, some limited support for QEMU consoles is available too. Within ctys the "-r" option is adapted from the "-g" option when not provided explicitly. As a consequence the resulting "-r" option is the maximum size which could be displayed without restarting the vncserver, shrinking is supported by pure clipping with additional display of scrollbars.

The other point to mention is the logical addressing of the screens only, which makes it independent from physical reassignments of display ports, e.g. due to a motherboard replacement.

In addition to the only required call of "ctys admin", any entity could be called easily as a single session by just an "smart-cut-and-paste". This is due to the containment of the complete set of options within the context, where each of them could be used as a self-contained set to replace the group name.

The start of "HOST3" could be written as:

```
ctys root@host3'( \
  -t vnc -a create=reuse,l:HOST3 \
  -g 1268x994:A00 \
  -D admin -b 1,2)'
```

26.2 Tax-Calculation and Longterm-Storage

The following example depicts the usage of VM and VNC sessions for the setup of the environment to calculate the taxes resulting from commercial activities.

The second point is the usage of VMs on the base of VMware in order to fulfill the required longterm archiving of the data and the software used for calculations. This is particularly handy, because only a compatible hypervisor is required later in order to reactivate the environment for later monitoring.

A QEMU based VM might possibly offer the most flexibility and best future compatibilty, due it's complete availability as open source and it's full support of execution within arbitrary levels of nested stacks. The only point to mention is the current requirement of a gcc-3.x version. The performance might no longer to be mentioned within a few years.

When considering 10years and more of required archiving, the dropped requirement to maintain the original hardware might be a point. From now looking backward this would be e.g. a i486 or p100 from 1997, which does not support a HDD bigger than some GiB.

The utilized and archived components coul be based on a linked VM, which eases the question of licencing, due to just openning multiple windows for smart usage of the applications.

In this example, which is a life example, the following components are used:

- SteuerSparerklaerung
  for tax calculations

- Lexware Financial Office Pro - Server-VM
  for calculations of the "statement of revenues and expenditures" - EUER
- Lexware Financial Office Pro-01 - Client-VM
  for display of the account information for input
- Lexware Financial Office Pro-02 - Client-VM
  for actual dialog input
- VNC-Server for central DB storage admin on Samba
  storage of centralized data base on a RAID array with Samba export to SMB clients.

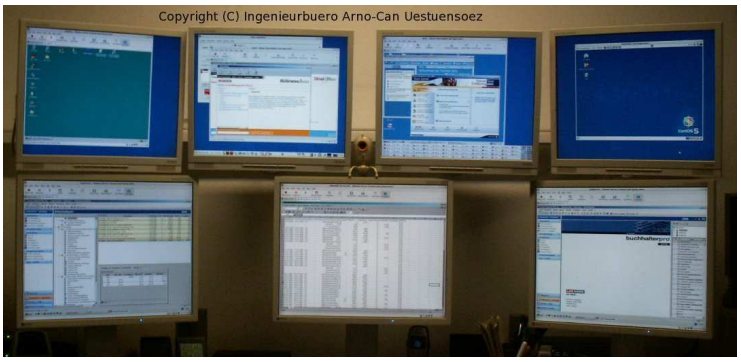

Figure 26.3: Registration and Longterm Storage of receipts for TAX requirements

In the previous scenario the screen A00 contains the Lexware server process running on a Windows-NT-4.0 Server with last updates SP6a. The hypevisor a VMware server. This is the only and one reason for keeping Windows-NT-4.0-S still alive.

The screens A10 contains several applications as "online-books" and databases containing information on taxes.

The screen A20 contains the tax calculations program for 2006.

The screen A30 is the dekstop for administrationaly tasks related to tasks such as backup on the hosting Linux-CentOS-5.0 server running Samba. The Linux version on this server is constantly updated, because it only has storage functionality.

The Screen A01, and A21 contain the client versions of Lexware server, where the left has opened the assests management modul and the right instance displays the bookkeeping module.

The center screen A11 contains an MS-Excel version, which is used to edit the records to be imported into the import queue of the bookkeeping. The usage of MS-Excel for a moderate number of receipts has the advance, that same types of receipts from same creditor could be just copied and require only to be adapted by the sum and date. Thus it is much faster for periodic payments than walking through the whole path of a dialogue.

Almost the same shortcut is applied as for the previous example. The minor difference results from a specific behaviour within this version. The prefetch values are read from the actual command line only, before macro expansion. Thus the utilized plugins have to be explicitly supported by call parameters for now.

ctys -T vnc,vmw tax2006

That's it.

The call consists of the mandatory base call "ctys" of course, and an additional pre-configure group called "tax2006".

```
#does not work within this version because of early
#pre-fetch by actual call only.
'(-T vnc,vmw )'

#
#This groups contains all machines for the
#tax2006 desktop.
#
root@delphi'(  \
 -t vnc -a create=reuse,l:TAXADMIN \
 -g 1268x994:A30 \
 -b 1,2)'
```

```
#server
tstusr@host07'( \
 -t vmw \
 -a create=reuse,p:/..../fop600s/fop600s.vmx \
 -g 1268x994:A00 \
 -b 1,2)'


#account references
tstusr@host07'( \
 -t vmw \
 -a create=reuse,p:/.../fop600c01/fop600c01.vmx \
 -g 1268x994:A01 \
 -b 1,2)'


#data registration
tstusr@host07'( \
 -t vmw \
 -a create=reuse,p:/.../fop600c02/fop600c02.vmx \
 -g 1268x994:A21 \
 -b 1,2)'


#tax calculation current year
tstusr@app2'( \
 -t vmw \
 -a create=reuse,p:/.../tax01/tax01.vmx \
 -g 1268x958:A10 \
 -b 1,2)'


#tax comparison last year
tstusr@app2'( \
 -t vmw \
 -a create=reuse,p:/...tax02/tax02.vmx \
 -g 1268x872:A20 \
 -b 1,2)'


#general admin office
tstusr@host07'( -t vmw \
 -a create=reuse,\
```

```
   p:/.../office001-01.01.001/office001-01.01.001.vmx \
-g 1268x994:A11 \
-b 1,2)'
```

26.3 QEMU Test-Environment for VDE

This example depicts the test environment for the first basic testst of the tool "ctys-setupVDE" . In this scenario PMs only are used to perform the following tests manually.

Screen	Machine	Hypervisor	Testcase
A00	host01	VMPlayer+QEMU	Single NIC without present bridge.
A10	host02	VMServer+QEMU	Bonding device only, without present bridge.
A20	-		
A30	host04	XEN+QEMU	Single NIC with a default bridge present.
A01	host05	XEN+QEMU	Bonding+Additional WoL-NIC, with present Xen bridge on bond0.
A11	-		
A21	host07	VM-WS6+QEMU	Bonding+Additional WoL-NIC, without present bridge.

Table 26.1: QEMU Base Tests for "ctys-setupVDE"

Due to the various possible and very common cases of present and recognisable virtual bridges, the history of LAN configuration met by "ctys-setupVDE" could span a number of different configurations. For each such case at least the "create" and "cancel" argument has to be tested and validated by checks via "brctl", ifconfig, and "route". Also multiple bridges for different users and multiple calls for the same user has to be performed.

The tests listed as example in the previous table just span the usage of PMs, that is native access to a machine. Additional tests are performed for stacks within GuestOSs on the various stack levels and OSs. The tests above are performed on

CentOS/RHEL-5.0 and additional systems.

However, the intention of this chapter is not to disclose test utilities and startegies, but to show one typical application of the UnifiedSessionsManager.

To open the following desktop with all of the previous listed scenarios the call

ctys tst-qemu

is sufficient.

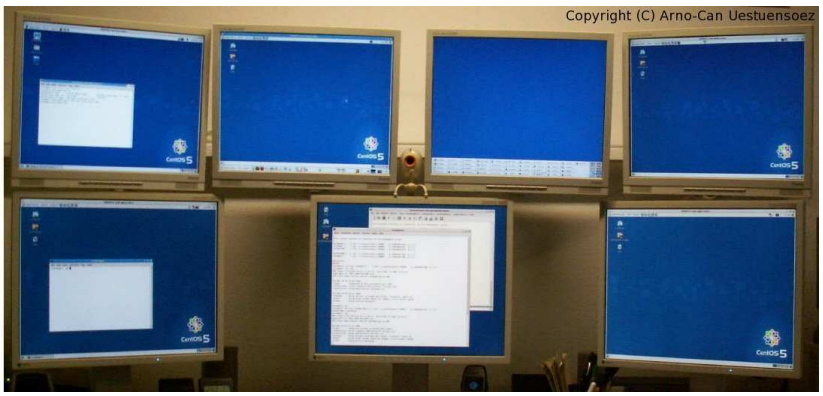

Figure 26.4: PMs for basic tests of "ctys-setupVDE"

That's it, with a little additional one-time pre-configuration.

```
root@host1'( \
 -t vnc \
 -a create=reuse,1:QEMU1 -g 1268x994:A01 \
 -b 1,2)'

root@host2'( \
 -t vnc \
 -a create=reuse,1:QEMU2 \
 -g 1268x994:A21 -b 1,2)'

root@host3'( \
```

```
 -t vnc \
 -a create=reuse,l:QEMU5 \
 -g 1268x958:A10 -b 1,2)'

root@host4'( \
 -t vnc \
 -a create=reuse,l:QEMU3 \
 -g 1268x994:A30 -b 1,2)'

root@host5'( \
 -t vnc \
 -a create=reuse,l:QEMU4 \
 -g 1268x994:A00 -b 1,2)'
```

Anyhow, the amount of tests required expands quickly with additional PMs.

Figure 26.5: Various PMs for expansion of test scope of "ctys-setupVDE"

Chapter 27

Performance Measures

27.1 Background, Parallel and Cached Operations

The following call deactivates the caching of data and operates in synchronous and sequential mode. This causes each target to be listed sequentially, which take for the actual test configuration about 72 seconds. This is caused particularly by the outdated machine lab01.

```
time ctys -a list=tab_tcp,id -C off -b seq,sync \
          lab00 lab01 app1 app2
```

The following call works similar, but activates the caching of, thus a result is displayed after processing all targets.

```
time ctys -a list=tab_tcp,id -C on -b seq,sync \
          lab00 lab01 app1 app2
```

When changing the "-b" option for altering the synchronism, the task duration is almost just the period required by the slowest machine, with minor overhead.

The following call now activates the parallel operation but still occupies the console caused by "sync" suboption.

```
time ctys -a list=tab_tcp,id -C off -b par,async \
          lab00 delphi olymp app1 app2
```

The overal required processing time is now aboout 30seconds, which is the individual time required by lab01. Without the lab01 it takes 17seconds as expected, which is 11seconds each with about 1-2seconds when bundling overhead.

The performance gain is in a linear manner, thus handling of 10 or more machines definetly benefits from these concept, which makes quickly a difference of 11-18 seconds vs. several minutes.

Even though the overall performance could be still enhanced, in comparison to manual handling of even 4 machines only with about 5-10 VMs of multiple types on each, this required processing time of 11-18seconds can not be blamed.

27.2 Caching of Data

The second main field of performance measures is the caching of static data for network nodes and VMs with their GuestOSs and hypervisors and caching some local runtime data of for repetitive access.

The caching of local runtime data is particularly helpful, when ID transformations for mapping and remapping of access keys has to be performed. This data has one-call life-period and is foreseen just for real short time caching in the range of 1second within a call.

The second area of caching is the local caching of network and VM data, which could be classified as static. This contains static configured DHCP data including the DNS entry, and the ENUMERATED VM configurations, including the information of their installed GuestOSs.

The whole set of local data is stored and managed by the tool ctys-vhost. Which will be used internally for cached querries for several kinds of name resolution. When the local cache database is not present ctys silently switches to polling mode, which is actually a "find-grep-scan" of the remote filesystem for

appropriate VM-configuration files. The performance impact of uncached operations depends on the call parameters and the dimension of the scanned directory tree, but could be immense.

When the data is appropriately cached the average access time of ctys-vhost to an VM record is within one second, for some hundred stored entities about 0.5-0.8 seconds. Which depends on several influencing parameters, of course. Anyhow, the performance gain is about a factor of 100 or more.

The local database has to be prepared by the tool ctys-vdbgen and ctys-extractMAClst and/or ctys-extractARPlst. Additionally a configuration file for the PM should be genrated by "ctys-genmconf".

27.3 Combined Operations

The generation of the caching database itself and the access to the cached data are the best examples for the comparison of cached and non-cached access to distributed work environments.

The draft tests are specific to the environment and some differences in the individual nodes, but they are representative.

In a test-group of 7 machines with a common NFS mounted pool, and a sum of 365 VM entries, which is about an average of 50 one-layer-VMs on each node, the following performance was measured.

Nr.	Access Method	"-b" option	duration
1	Collection of data with ENUMERATE	sync,sequential	4:16
2	Collection of data with ENUMERATE	sync,parallel	1:14
3	Query ctys-vhost, first access with rebuild cache	n.a.	1:53seconds
4	Query ctys-vhost for all accespoints of "linux001"	n.a.	0.6seconds

Table 27.1: Performance effects of "-b" option

In a production-group of 9 machines with a common NFS mounted pool, multiple users with deep home-directory-trees, and a sum of 1768 one-layer-VM entries, which is about an average 200 VMs on each node, the following performance was measured.

Nr.	Access Method	"-b" option	duration
1	Collection of data with ENUMERATE	sync,sequential	15:26
2	Collection of data with ENUMERATE	sync,parallel	3:56
3	Query ctys-vhost, first access with rebuild cache	n.a.	2:06
4	Query ctys-vhost for all accespoints of "linux001"	n.a.	0.8seconds

Table 27.2: Performance effects of "-b" option

Chapter 28

VM Stacks

28.1 Setup of the Stack Environment

In current version the stacked operation of more than 2 nested layers is primarily for test reasons. This is going to be changed for daily business step-by-step, whereas stack-aware appliances are developed.

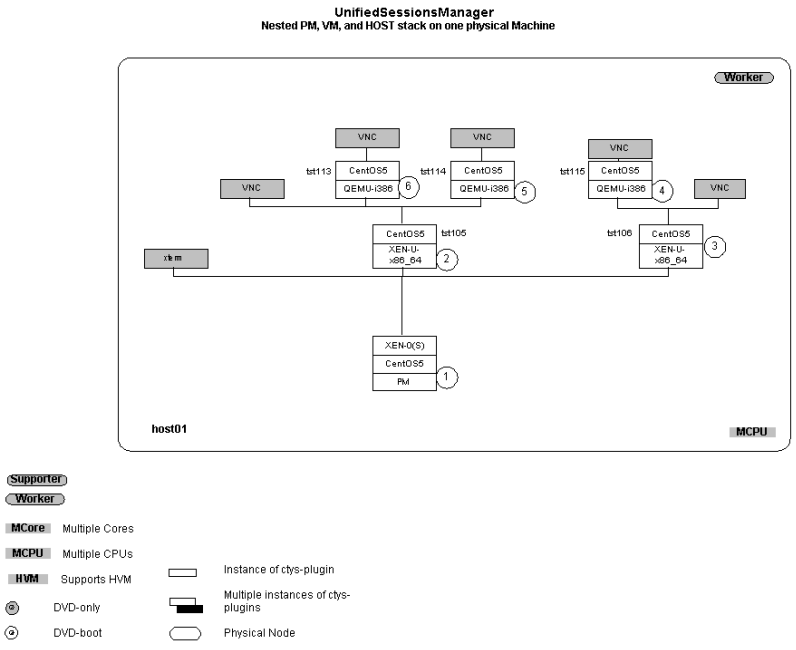

Figure 28.1: Example for a simple Stack

The standard case for now is up to 2 layers for the combination of QEMU with Xen and/or VMware hypervisors. In some cases a combined network simulation group is setup into one VM, thus could be frozen and debugged as one unit when required.

The following architecture is a common operational environment for a small network, which already is designed Service-Centric by specific Stack-Components.

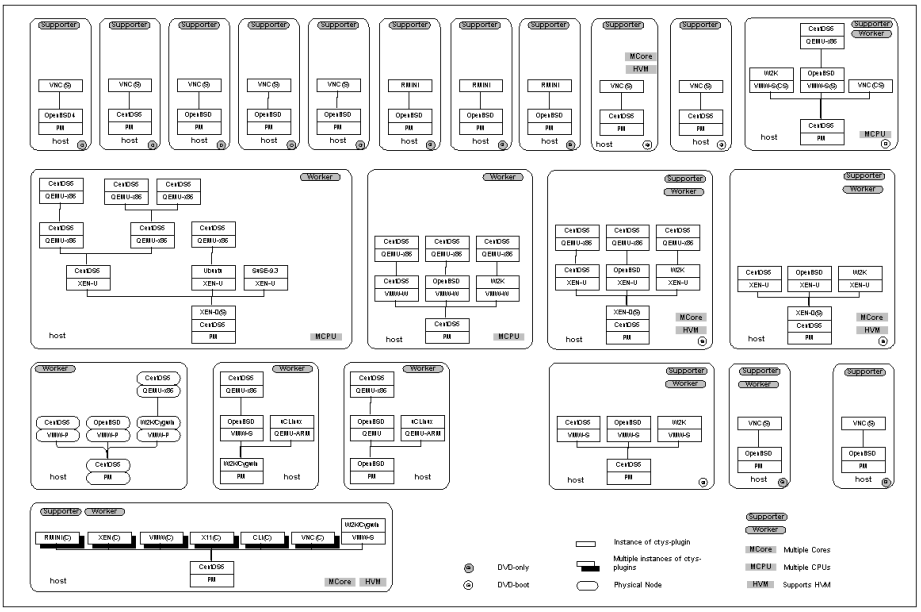

Figure 28.2: Example for a Small Stacked-Network

For now the author is looking forward for the arrival of the first announced 6-core and 12-core CPUs by AMD and for sure by Intel too, meanwhile preparing it's software appropriately by usage of 2-core and 4-core CPUs.

The main building block for nesting of stacks available for now is QEMU as a pure generic emulator. Therefore as already mentioned the KQEMU module is avoided. For usage of VM images in general, but nested stack elements - here QEMU - in

particular, the setup of a common network structure for neatless usage within each participating entitiy is essential. This for example enables the usage of the same VM within multiple stack levels and within various VMs. Thus a common filesystem hierarchy is defined based on NFS for VM distribution and nesting.

For performance critical cases an additional local structure is set up for enhancing acces to stored data.

The second issue for distribution and nesting of VMs is a SSO - Single-Sign-On - authentication and authorization. This is required for the actual payload users within the GuestOSs as well as for the sessions management of the hypervisors by UnifiedSessionsManager itself. The choice for the examples and the actually installed environment is a combination of Kerberos, LDAP, Samba, NFS, and Automount/Autofs. This is particularly combined with "ksu" and "sudo" for the internall call of system utilities by ctys.

For the given examples the following structure is defined as basis, which is used within all VMs and PMs. Therefore each custom script and filepath could be used within each VM, which could be instanciated independent of it's execution environment.

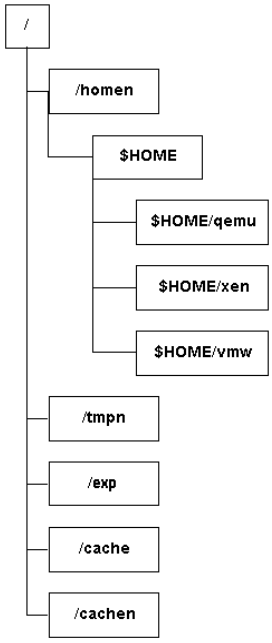

Figure 28.3: Filesystem Structure

The usage of NFS as the global storage requires an appropriate network performance of course. Thus bonding of GiB lines is widely utilized, what offers a quite well throughput with limited costs. This is recommended for central servers and heavily loaded application servers homing more than 10-20 VMs, what depends on the actual load of each.

The physical structure of distribution within a network is as depicted in the following figure.

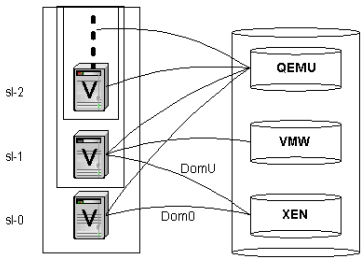

Figure 28.4: Stacked VMs and Filesystem Access

The network contains the generic VMs which could be relocated to various execution sites and levels. The only available vertically arbitrarily stackable VM is QEMU, whereas the VMW

and XEN hypervisors require a "dominant" ownership of all present CPUs and could be used for current versions EXOR only on a single machine.

28.2 Single-Upper-Layer VMStack Basics

28.2.1 CREATE a PM Session

For flexible usage of physical machines each participating node should be configured for activation by WoL. This allows flexible load concentration and distribution. The UnifiedSessionsManager for now supports Linux and OpenBSD based machines, and offers functionality for setting dynamic the WoL flags as well as sending a wakeup message.

28.2.2 CREATE a VM Session with initial PM Session

28.2.3 CREATE a HOST Session with initial VM and PM Session

28.2.4 CANCEL a VM Session containing VM-Stack-Sessions

28.2.5 CANCEL a PM Session containing VM-Stack-Sessions

28.3 Multi-Layer VMStacks

28.3.1 MACRO Definition Basics

Common Syntax

Consoles and Synchronous STACK-Operations

Integration of Tool-Calls

Auto-Creation of Virtual Bridges and Switches

28.3.2 CREATE a VMStack

Create a Blue-Print of Stack-Data

Verification of Supported Actions and Modes

Verification of Stack Entity-Consisytency

Verification of Stack-Capability

Verification of Hardware-Capability

Verification of Stack-Location

28.3.3 CANCEL a VMStack

28.3.4 Prepare a PM for Restart by WoL

This call prepares the physical machine for wakeup by WoL packets.

```
#
#
#First version of stack-aware tests, with canonical syntax.
#
#

##########################################################
#
#generic preparation of a machine for WoL restart
#
tst-prep    = \

  #
  #local options
  #
  -t pm \
  -a cancel=poweroff:0,force,self,wol \
  -b 1,3 \

  -- \

  #
  #remote-only options
  #
  '(-Z ksu)'
```

28.3.5 Xen with upper QEMU-VMs

This example shows the creation of one "column" of a VM-Stack, where 3 Stack-Layers are created.

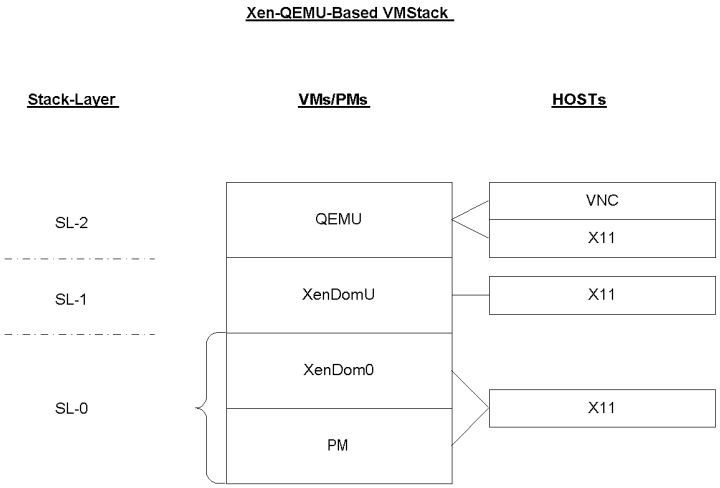

Figure 28.5: Example for a Single Stack-Column

The following virtual network components are involved in the interconnection of the various VM-Stack-Layers to the physical NIC and though to the actual physical network.

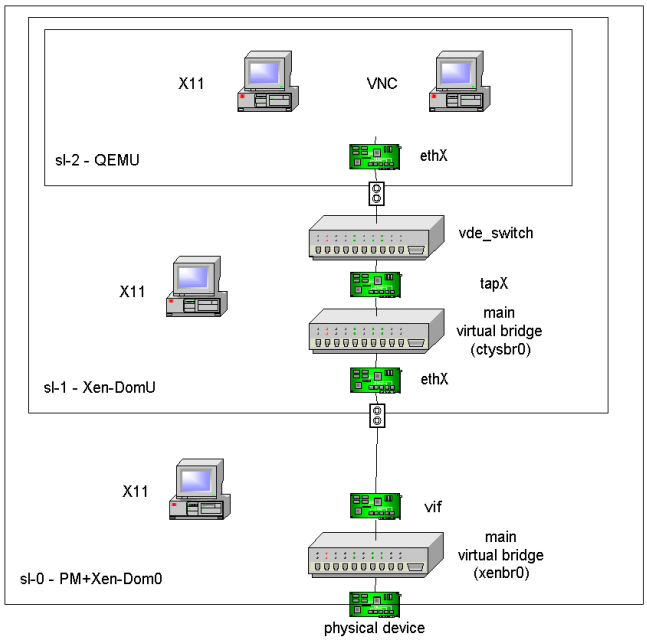

Figure 28.6: Example for a Single Stack-Column of Network Components

The following example is the literal content of a specific MACRO for test purposes. This particularly includes the remote call of "ctys-setupVDE" for remote and automated creation of the virtual bridge and swithc within the "SL-1".

```
############
#
#  XEN+QEMU: app1
#
#  Description:  First successful stack-start
#
tst-01a = VMSTACK'{\


 ########################################################
 #1.) SL-0: Relay for WoL on base PM
 ########################################################
 ws1( \
   -t pm  \
   -Z ksu \
   -a create=l:app1,t:app1-eth2,m:00:E0:81:2B:10:33,wol,reuse \
   -c off \
 ) \


 ########################################################
 #2.) SL-0: Creation of native acces by xterm.
 ########################################################
 app1( \
   -t x11 \
   -Z ksu \
   -a create=l:STACK-APP1,console:xterm \
   -g 80x13+0+800:A00 \
 ) \


 ########################################################
 #3.) SL-1: First level VM, a Xen-DomU here
 ########################################################
 app1( \
```

```
  -t xen \
  -Z ksu \
  -a create=l:tst104,reuse,console:none \
  -c local \
) \


#########################################################
#4.) SL-1: Creation of native acces by xterm.
#########################################################
tst104( \
  -t x11 \
  -Z ksu \
  -a create=l:TST104,console:xterm \
  -g 80x13+0+600:A00 \
) \


#########################################################
#5.) SL-1: Creation of required TAP device and a
#          bridge when missing, a switch for
#          non-privileged USER-SPACE usage by user.
#########################################################
root@tst104( \
  -t cli \
  -Z ksu \
  -a create=l:tstVDE,cmd:ctys-setupVDE%-f%-u%acue%create \
) \


#########################################################
#6.) SL-2: Creation of second-level VM, QEMU here.
#########################################################
tst104( \
  -t qemu \
  -Z ksu \
  -a create=l:tst127,reuse,console:none,user:wol1 \
  -c local \
) \


#########################################################
#7.) SL-2: Creation of native acces by xterm.
```

```
 ########################################################
 wol1@tst127( \
   -t x11 \
   -Z ksu \
   -a create=l:TST127,console:xterm \
   -g 80x13+0+400:A00 \
 ) \


 ########################################################
 #8.) SL-2: Creation of native acces by VNC.
 ########################################################
 wol1@tst127( \
   -t vnc \
   -Z ksu \
   -a create=l:mydsk,reuse \
   -g 400x300+0+0:A00 \
 ) \


}'
```

The resulting display should be as follows, some hostnames has to be adapted of course.

Figure 28.7: VMSTACK tst-01a in action.

The list call

```
list '{macro:listjobs}' app1 tst104 wol1@tst127
```

should result in an output similar to the following.

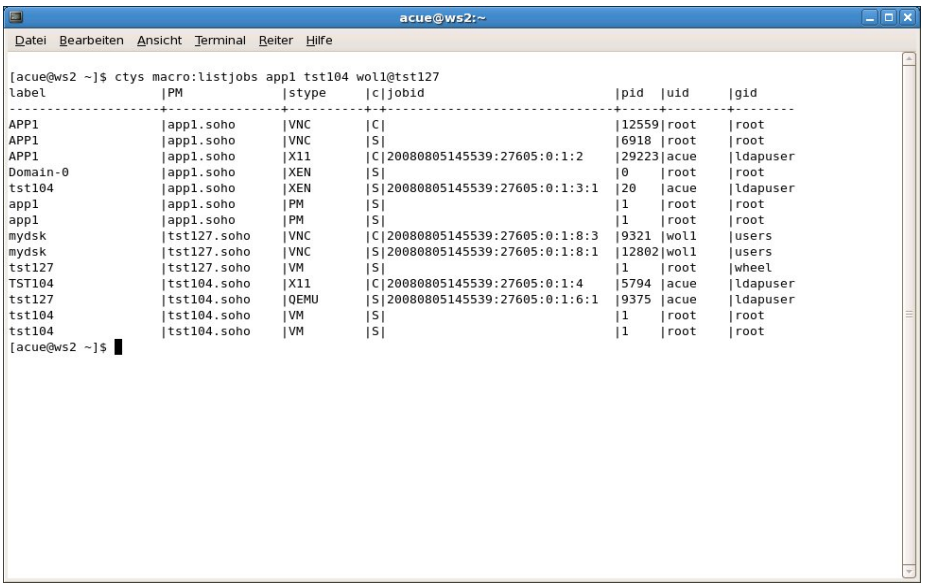

Figure 28.8: LIST of running VM instances

As displayed, the job data of the user root is visible only to root. Currently this feature is under development, thus not yet consequently applied to all plugins.

28.3.6 VMWare with upper QEMU-VMs

28.3.7 QEMU with upper QEMU-VMs

Chapter 29

User Access - Secure permissions

29.1 SSH, SSL, IPsec, and VPNs

The only supported remote access method of ctys-tools is the usage of SSH, any other method might be used in companion.

Even though SSH keys could be utilized and some other methods like r-files could be used, these are not discussed here.

The only and one SSO approach used and supported by the author is a kerberos and LDAP based SSO in companion with kerberised SSH.

With the usage of automount NFS(will be changed soon to AFS), a seamless authorisation and environment support throughout stacked VMs is supplied.

For root access a local account should remain.

29.2 VM-Stacks, SSO, and Kerberos

29.3 Network Filesystems

29.4 Authentication - ksu and sudo

29.4.1 Basics

Due to the timeout behaviour of ksu and sudo during probing when no user is configured, the default behaviour is not to probe for theese tools.

By default the user permissions are used only, thus any systems maintenance operations could be performed by usage of the root user, or any other with root permissions.

This could be utilized for example by usage of k5login with kerberos.

When ksu or sudo has to be used, which means internally probed for access permissions, this has to be preconfigured by the environment variables "USE_SUDO" and/or "USE_K5USERS". Alternatively this could be set call-by-call and different for local and remote components with the option "-Z". When using "sudo" it has to be decided whether pty is required or not, if yes, than the "-z" options has to be set.

29.4.2 sudoers

Sudoers is some more flexible adaptable than k5users, it particularly allows access-profiles by distinguishing several call-options, e.g. CANCEL, CREATE, etc.. The approach of lean call options in difference to supplying more action keys on first level-options just makes it little more challenging, but not impossible.

This is particularly helpful when restricting access selectively to specific call options. E.g. in case of Xen to the required Dom0 for LIST action call.

When using sudoers the following entries are required for this version.

The users as required:

User_Alias CTYS_USER = chkusr, wol1

At least one user with complete local acces is required, when a bridge has to be shutdown for WoL setting, and this is the only LAN connection to the caller. Thus during the shutdown the NFS mounts are on those machines out of service for a short time where additional calls to scripts are reuqired. So the shutdown could be performed from a local copy only.

If not tty should be pre-required for sudo. If the default remains, the "-z" option is required.

Defaults requiretty

For machines without required bridges the following permissions are required:

```
Cmnd_Alias CTYS_CMD = \
  /usr/bin/which \
  ,/sbin/halt ,/sbin/ethtool ,/sbin/reboot ,/sbin/init
  ,/sbin/poweroff
```

For machines to be used to send WoL packets to local segment addtionally the following permissions are required:

```
Cmnd_Alias CTYS_CMD = \
   /sbin/ether-wake
```

When a bridge is configured on the target, which has to be shutdown for setting "wol g", which is required for Xen-3.0.2, following root permissions are reuqired

```
Cmnd_Alias CTYS_CMD = \
   ,/etc/xen/scripts/network-bridge ,/sbin/ip \
   ,/usr/sbin/brctl ,/sbin/ifup ,/sbin/ifdown \
   ,/usr/bin/head
```

When running Xen on local machine the following has to be added:

```
Cmnd_Alias CTYS_CMD = \
   ,/usr/sbin/xm ,/usr/bin/virsh \
```

For the users assignement an entry like this has to be set:

```
CTYS_USER ALL = (root) NOPASSWD: CTYS_CMD
```

29.4.3 k5users

For a present kerberos environment the utilization of krb5users seems to be the smartest approach, even though the call itself cannot be matched by specific CLI-options. This lack for selective access profiles could be worked around by defining different users and appropriate chroot-environments easily.

For machines without required bridges the following permissions are required:

```
<user>@<realm> \
  /usr/bin/which \
  /sbin/halt  /sbin/ethtool /sbin/reboot /sbin/init \
  /sbin/poweroff
```

For machines to be used to send WoL packets to local segment addtionally the following permissions are required:

```
   /sbin/ether-wake
```

When a bridge is configured on the target, which has to be shutdown for setting "wol g", which is required for Xen-3.0.2, following root permissions are reuqired

```
   /etc/xen/scripts/network-bridge /sbin/ip \
   /usr/sbin/brctl /sbin/ifup /sbin/ifdown \
   /usr/bin/head
```

When running Xen on local machine the following has to be added:

```
/usr/sbin/xm /usr/bin/virsh
```

29.4.4 Secure root Access

If required the k5login could be used for ctys. But this should be used on test machines only.

29.5 Firewall

For acces by ctys the only port to be opened is the ssh=22 port. Any communications for DisplayForwarding and COnnection-Forwarding is processed by ssh.

When setting up a router for remote WoL some configuration for sending Ethernet-Broadcasts or a directed IP-broadcast is required.

29.6 SELinux

Chapter 30

System Resources

30.1 TAP/TUN by VDE

A TAP device in combination with virtual switches will be applied in accordance to the introduction within Section 8.3 'Stacked Networking' on page 95 . The actual implementation is based on the package *"Virtual Distributed Ethernet" - VDE(vde2)*[130, sourceforgeVde].

The only and main restriction was the originally required presence of the package "userspace-utils" from the UML project, containing the "tunctl" tool for tap creation and assignemt of it's owner. This is, at least not out-of-the-box available for the majority of distributions, but within *"Virtual Distributed Ethernet"*, by the time of writing this dokument it is not yet documented.

The tool called "vde_tunctl" from *"Virtual Distributed Ethernet"* could be used as a perfect replacement. The additional benefit is the included "vde_switch", which is a virtual switch, able to operate in hub-mode. This supports user-space access and on demand creation of new interfaces with non-privileged users permission only. Just one initial TAP device is required to be created with root permissions.

Thus the following basic structure is assumed to be pre-configured for each user and interconnected to the main virtual-bridge:

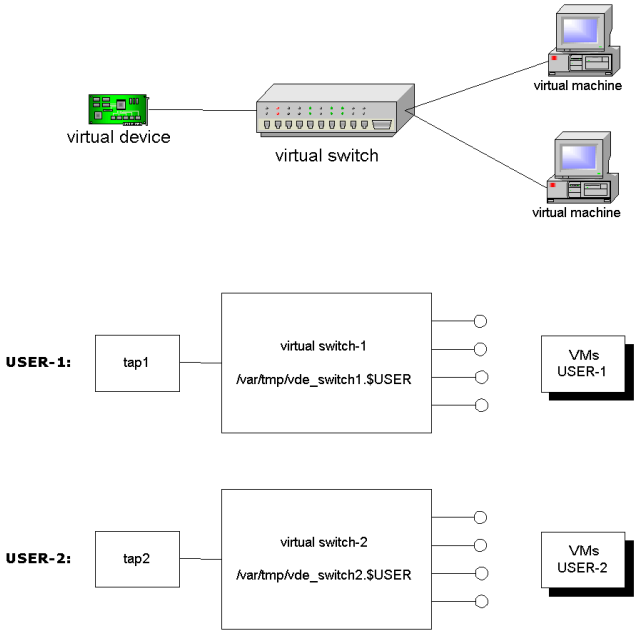

Figure 30.1: Virtual switch by vde_switch

Thus each user allowed to access external network has it's individual virtual switch with user-space access.

30.1.1 VDE within Xen Dom0

This lists required steps to setup a QEMU instance within a Xen-Dom0 VM using the default xenbr0 bridge for external interconnection. The most of the required steps are covered within ctys-setupVDE . Due to required root-permissions for some of called system tools appropriate sudo/ksu configuration should be in place. For the display of a listing of called tools refer to Section 18.8 'ctys-plugins' on page 352 .

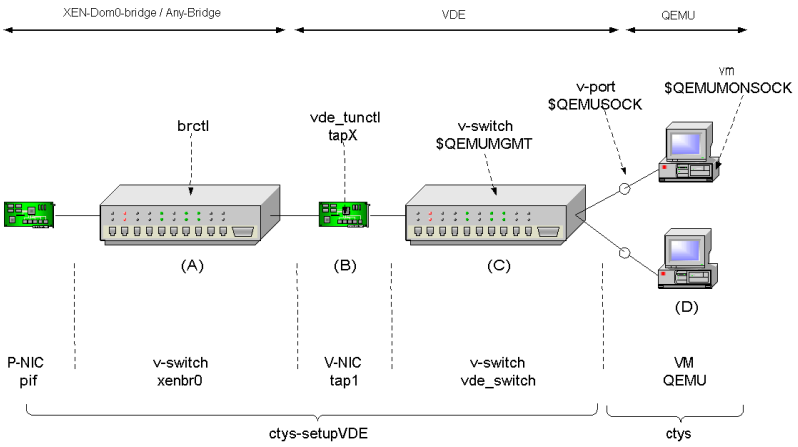

Figure 30.2: QEMU NIC interconnection

1. For *"CentOS-5.0"*[74, centOS] the *"vde2-2.2.0-pre1 version"*[130, sourceforgeVde] seems to work perfectly.

   Just download [130, sourceforgeVde] it and follow the instructions. For the original distribution a "configure+make+make install" is required.

2. Prepare networking for Qemu by calling

   ```
   ctys-setupVDE -u <switch-owner> create
   ```

   This call creates a new switch to be used e.g. by QEMU, where <switch-owner> is the user which owns the communications sockets to the switch, default is "$USER".

   Even though it is possible to use sudo and/or ksu for ctys-setupVDE , this is not recommended, because e.g. chmod and chown are reuquired in order to change the owner of originally root-owned switch sockets. Thus the creation of the bridge and user-switches should be performed by the administrator as root, or as a admin-only account.

   This tool manages the complete network infrastructure required for QEMU and creates required bridges and switches as required. Thus the user does not need to care for the preconfiguration of the network resources.

The following network configurations are supported for "bridging" external NICs.

- single NIC, without a preconfigured bridge
- single NIC, with a preconfigured bridge, e.g. by Xen
- multiple interfaces as a bonding devide to be bridged
- multiple interfaces as a bonding devide with a proeconfigured bridge, e.g. by Xen

The ctys-setupVDE , utility can be applied within Xen-Dom0, Xen-DomU, VMware, QEMU, and native; additional options are available.

3. Call QEMU with a VDE wrapper
   The final call from within ctys should be wrapped in accordance to internal call interface, the "raw" call shown here is just for an initial test by provided QEMU-Example VMs. ctys-wrapper for QEMU-Examples are provided within the installed "$HOME/ctys/templates/qemu-examples-wrapper" subdirectory.

   The variable QEMUSOCK could be configured within qemu.conf.$MYOS.

   ```
   QEMUSOCK=/var/tmp/vde_switch0.$USER

   vdeqemu -vnc :17 \
       -k de \
       -net nic,mac=<MAC-addr> \
       -net,sock=${QEMUSOCK} \
       linux-0.2.img \&
   ```

4. Attach a vncviewer

   ```
   vncviewer :17&
   ```

5. Connect GuestOS to external network

   ```
   ifconfig eth0 <IP-address>
   ```

   The default address is something like "10.0.2.15", when your network fits to this, nothing is required to be done.

6. Call ssh, and or ping, or whatever you want.

That's it.

30.1.2 VDE within Xen DomU

See Section 30.1.1 'VDE within Xen Dom0' on page 580 .

30.1.3 VDE within VMware

See Section 30.1.1 'VDE within Xen Dom0' on page 580 .

30.1.4 VDE within Native Unix

These are the required steps to setup a QEMU instance on a native Linux system creating a new bridge with bridctl for external interconnection. The most of the required stecps are covered within ctys-setupVDE . Due to required root-permissions for some of called system tools appropriate sudo/ksu configuration should be in place. For the display of a listing of called tools refer to Section 18.8 'ctys-plugins' on page 352 .

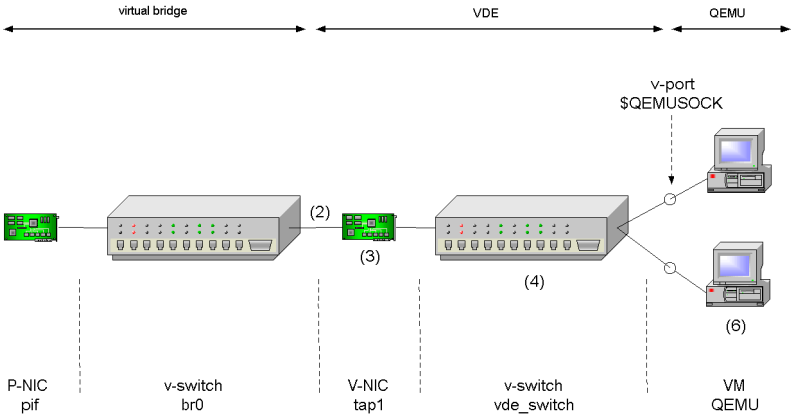

Figure 30.3: QEMU NIC interconnection for Native UNIX

Pre-required admin tasks with root permissions, not called within ctys.

1. For *"CentOS-5.0"*[74, centOS] the *"vde2-2.2.0-pre1 version"*[130, sourceforgeVde] seems to work perfectly.

   Just download it and follow the instructions. For the original distribution a "configure+make+make install" is required.

2. Create: root permission required
   "tapX" with
   `"vde_tunctl -u <user-without-root-permission>"`
   e.g.
   `vde_tunctl -u acue`
   Returns a line like:
   `"Set 'tap3' persistent and owned by uid 4711"`
3. Call: root permission required
   Create a new bridge:
   `brctl addbr br0`
4. Call: root permission required
   Disable STP for now:
   `brctl stp br0 off`
5. Call: root permission required
   Add interface - bonding interface here:
   `brctl addif br0 bond0`
6. Call: root permission required
   Deactivate bonding interface:
   `ifconfig bond0 down`
7. Call: root permission required
   Set to promiscous mode:
   `ifconfig bond0 0.0.0.0 up`
8. Call: root permission required
   Set IP address for host:
   `ifconfig br0 <ip-of-bond0> up`
   Call ping, but may take some trials and some seconds - 4-5 fails within about 20-40seconds, than works.
9. Call: root permission required
   Use the returned 'tap3' for following call:
   `/etc/xen/qemu-ifup tap3 br0`
   This is due to the usage of QEMU within Xen, for the targeted stack test. Any other configuration will be similar to this.

   The given script could be at various locations, it contains basically the following calls only:
   ```
   ifconfig $1 0.0.0.0 up
   brctl addif $2 $1
   ```
   Which brings up the newly created interface 'tap3' and adds an interface to the virtual Xen bridge connecting it to the world outside.

The results could be verified with:

ifconfig tap3
should list an interface 'tap3'

brctl show
should contain an interface 'tap3'

10. Call: root permission required
Now this interface will be connected to another virtual switch, the vde_switch in order to brovide an internal multiplexer for multiple QEMU instances to be connected to the external interfaces via the Xen-bridge.

```
QEMUSOCK=/var/tmp/vde_switch0.$USER
QEMUMGMT=/var/tmp/vde_mgmt0.$USER

vde_switch -d \
    -tap tap3 \
    -s ${QEMUSOCK} \
    -M ${QEMUMGMT}


chown -R <userX.groupX> ${QEMUSOCK}
chown -R <userX.groupX> ${QEMUMGMT}
```

11. Call: optional

```
QEMUMGMT=/var/tmp/vde_mgmt0.$USER

unixterm ${QEMUMGMT}
```

User tasks, not requiring root permissions, called within ctys.

12. Call:
The variable QEMUSOCK could be configured within qemu.conf.$MYOS.

```
QEMUSOCK=/var/tmp/vde_switch0.$USER

vdeqemu -vnc :17 \
    -k de \
    -net nic,mac=<MAC-addr> \
    -net,sock=${QEMUSOCK} \
    linux-0.2.img \&
```

13. Call:

```
vncviewer :17\&
```

14. Call within GuestOS:

```
ifconfig eth0 <IP-address>
```

The default address is something like "10.0.2.15", when your network fits to this, nothing is required to be done.

15. Call ssh, and or ping, or whatever you want.

That's it.

30.1.5 VDE Remote Configuration

The tool ctys-setupVDE . supports the remote configuration and management of virtual VDE based switches. Due to the supported call with a list, multiple targets could be configured by one call. The required access permissions need to be pre-configured of course.

The following call creates on the hosts: "host1, host2, host3" a VDE-Switch and if not yet present a "Main Virtual-Bridge" (see figure: 8.9). The user is as default the caller's ID, here "root".

```
ctys-setupVDE create root@host1 root@host2 root@host3
```

The same could be perfomred by any user, but requires than for creation release access to system resources for the following tools(paths may vary) either by ".k5users" or "sudoers":

- /usr/local/sbin/vde_tunctl
- /sbin/ifconfig
- /usr/sbin/brctl

Than the same call e.g. looks like:

```
ctys-setupVDE \
  -u charly \
  create \
  host1'(-Z ksu)' \
  host2'(-Z sudo)' \
  host3'(-Z ksu)'
```

The following call lists all present switches on all hosts, including the dynamically allocated sockets.

```
ctys-setupVDE listall host1 host2 host3
```

The following call lists all present switches on all hosts, excluding the dynamically allocated sockets.

```
ctys-setupVDE list host1 host2 host3
```

The following call lists the ports of the switches owned by "root".

```
ctys-setupVDE ports root@host1 root@host2 root@host3
```

The following call lists the ports of the switch owned by "charly".

```
ctys-setupVDE \
  ports \
  -u charly \
  root@host1 \
  root@host2 \
  root@host3
```

The following call displays some basic information.

```
ctys-setupVDE info root@host3
```

The following call removes the previously created switches and eventually created "Main Virtual-Bridges", if they are no longer in use.

```
ctys-setupVDE \
 create \
 root@host1 \
 root@host2 \
 root@host3
```

Due to the remote call support of the tools ctys-genmconf and ctys-plugins the configured information could be verified for the various plugins easily. The required calls are:

ctys-plugins -T all -E root@host1 root@host2 root@host3

and for actual modification of the PM, either actually a physical machine

ctys-genmconf -P PM root@host1 root@host2 root@host3

or a virtual stack-entity, but appearing as physical for it's containees

ctys-genmconf -P VM root@host1 root@host2 root@host3

.

30.1.6 VDE-Setup with Micro-VMSTACK

The setup procedure of the required network environment based on "ctys-setupVDE" could be automated by usage of a "micro-VMSTACK" by defining either a macro or a synchronous subgroup and/or STACK mode. This is based on the remote-execution-call feature of the standard plugin CLI , which supports the remote execution of library functions and external executables. The reason for the usage of CLI as an excution framwork instead of the remote call capability of "ctys-setupVDE"(which is based on CLI), is simply the neatless integration of STACK aware synchronity, which is supported by the framework for plugin calls within ctys only.

A typical application of this feature within a full sized VM-STACK is shown in the Section 28 'VM Stacks' on page 559 .

```
completeQEMUExample = VMSTACK'{\
```

```
 ########################################################
 #0.) Synchronous pre-creation of required TAP device
 #    for a QEMU instance.
 #    A bridge is created when missing, additionally
 #    a switch for non-privileged USER-SPACE usage
 #    by user.
 ########################################################
 root@tst104( \
   -t cli \
   -Z ksu \
   -a create=l:tstVDE,cmd:ctys-setupVDE%-f%-u%acue%create \
 ) \


 ########################################################
 #1.) Creation of a VM based on QEMU.
 ########################################################
 tst104( \
   -t qemu \
   -Z ksu \
   -a create=l:tst127,reuse,console:none,user:wol1 \
   -c local \
 ) \

}'
```

30.2 Mount an ISO-Image

Loopback on Linux

Mount an FDD image (from "man mount"):

```
mount /tmp/fdimage /mnt \
  -t msdos \
  -o loop=/dev/loop3,blocksize=1024
```

Mount an CD/DVD image

```
mount <ISO-image-pathname> <mount-point> \
   -t iso9660 \
```

```
-o loop[=/dev/loop<free #dev>[,blocksize=1024]]
```

When "-o loop" provided only, the system will use appropriate defaults. The actual values could be inspected by "mount", the "losetup" call inspects loop devices.

Loopback on OpenBSD

Chapter 31

Applications of ctys-vhost

31.1 Database Generation ctys-vdbgen and companions

The cache database required by "ctys-vhost" is generated by the call of "ctys-vdbgen" , which internally simply calls "ctys -a ENUMERATE" . The output is literally stored into a first-level cache and correlated with additional information to a cacheDB for performance reasons.

The resulting performance gain is simply more than 100-1000 for the majority of practical cases.

The combination of the GROUP feature with "ctys-vdbgen" could be utilized to get smart calls and the build of different views as independent name-resolution subsets controlled by the "-p" option.

Anyhow, the presented set of functionality is mainly covering the collection of data from distributed VM stacks, including append of data to existing databases. The usage of multiple databases is covered. A temporary database sould be created, while the main is still processed, the later update simply requires a copy of the "enum.fdb" file, and a call to ctys-vhost(without the "-s" supress update option).

The management of the data could be simply done by any ASCII editor or any spread-sheet application like from OpenOffice or Microsoft(TM) MS-Office.

31.1.1 Initial Database

The following call generates the cacheDB for the user "tstusr" by usage of a predefined GROUP .

```
time ctys-vdbgen $USER-all
```

It is recommended to use the depicted trace call for a reduced and easy understandable progress tracking, which should be stored within an eventually prepared macro file.

```
time ctys-vdbgen $USER-all'(-d 2,w:0,s:16,p)'
```

For additional description of the utilized trace options refer to "-d" common option.

The following data is initially displayed.

```
Prepare execution-call:

Require DB-PATH,        USE: -o => "/homen/acue/.ctys/db/tststate1"
Stripped CLI            CLI:    => " tst-subgroup-00(-d 2,w:0,s:16,p)"
APPEND mode off => REPLACE : OFF(0)
STDIO mode off             : OFF(0)
Set TYPE scope          ADD: DEFAULT="-t ALL"
Preload TYPE set        ADD: DEFAULT="-T ALL"
Should speed-up         ADD: DEFAULT="-b sync,parallel "
Nameservice cache       OFF: DEFAULT="-c off "
Data cache              OFF: DEFAULT="-C off "

Resulting ENUMERATE     ADD: DEFAULT="-a enumerate=matchvstat:active,\
   machine,b:~%/etc/ctys.d -C off -c off -b sync,parallel \
   -T ALL -t ALL  tst-subgroup-00(-d 2,w:0,s:16,p)"

RESULTING              CALL:"ctys -a enumerate=matchvstat:active,\
   machine,b:~%/etc/ctys.d -C off -c off -b sync,parallel -T ALL \
   -t ALL  tst-subgroup-00(-d 2,w:0,s:16,p)>.../db/tststate1/enum.fdb"

-> generate DB(may take a while)...
-----------------------------------
START:19:50:30
------
ctys:acue@ws2.soho:1106:lib/groups:548:INFO:1:checkAndSetIsHostOrGroup:\
    delayed member expansion to next level for \
    SUB-GROUP=tst-subgroup-00-ws1
ctys:acue@ws2.soho:1106:CORE/EXEC:628:INFO:1:finalizeExecCall:delayed\
    member executed as SUBGROUP=acue@tst-subgroup-00-ws1
ctys:acue@app2.soho:28179:ENUMERATE/enumerate:234:16:2:scan4sessions:\
    START=19:50:46
ctys:acue@app2.soho:28179:ENUMERATE/enumerate:240:16:2:scan4sessions:\
    START-CLI=19:50:46
ctys:acue@app2.soho:28179:ENUMERATE/enumerate:249:16:2:scan4sessions:\
    FINISHED-CLI=19:50:46
ctys:acue@app2.soho:28179:ENUMERATE/enumerate:250:16:2:scan4sessions:\
    DURATION-CLI=00:00:00
...
...
```

Figure 31.1: TAB_CPORT by ctys-vhost

After this several minutes could be required for the now ongoing and probbable large ENUMERATE joblist. For this version no progress indicator is given.

During the execution of the job(s) several output may be presented, which gives some hints in case of erroneous execution. The following data is initially displayed.

```
Warning: No xauth data; using fake authentication data for
   X11 forwarding.
/usr/bin/xauth:  error in locking authority file
   /homen/vadmin/.Xauthority
X11 connection rejected because of wrong authentication.
X11 connection rejected because of wrong authentication.
```

Figure 31.2: TAB_CPORT by ctys-vhost

Once the job is completed, the final message is shown.

```
------
RET=0
------
...finished.

real    3m45.744s
user    0m3.880s
sys     0m4.260s
```

Figure 31.3: TAB_CPORT by ctys-vhost

As depicted, the job took about 4minutes. The results of the test example included here 19 remote accounts, on 8 machines and leads to 1297 VM configurations including fully functional VMs. As listed for first analysis of the cached content.

```
ctys-vhost -S list
```

This shows the complete cache, including the group definitions.

```
Current file-databases for "tstuser":
8k  160 /homen/tstuser/.ctys/db/default/macmap.fdb
248k  1297 /homen/tstuser/.ctys/db/default/enum.fdb


Current group files of:

CTYS_GROUPS_PATH   = /homen/tstuser/.ctys/groups
:/mntn/rd/p-int/tools/ctys/src/ctys.01_06_001a09/conf/ctys/groups


/homen/tstuser/.ctys/groups
4k   19/0      19  tstuser-all
4k   19/0      19  tstuser-test
4k   78/0      78  admin
4k   27/0      27  allVMs
4k   11/0      11  errgrp1
4k   26/0      26  test-lab
4k    2/0       2  tstgrp1
4k    9/0       9  tstgrp2
4k   55/0      55  tst-qemu

/mntn/rd/p-int/tools/ctys/src/ctys.01_06_001a09/conf/ctys/groups
4k    2/0       2  tst-001
4k    6/0       6  tst-002
4k    4/3      11  tst-nest-000-a00
4k    2/1       3  tst-nest-000-a10
4k    2/1       3  tst-nest-000-a11
4k    1/0       1  tst-nest-000-a12
4k    1/0       1  tst-nest-000-a20
4k    4/3      23  tst-nest-001-a00
4k    3/2       7  tst-nest-001-a10
4k    2/1       5  tst-nest-001-a11
4k    3/2       7  tst-nest-001-a12
4k    1/0       1  tst-nest-001-a20
4k    2/1       3  tst-nest-001-a21
4k    2/1       3  tst-nest-001-a22
4k    2/1       3  tst-nest-001-a23
4k    1/0       1  tst-nest-001-a24
4k    1/0       1  tst-nest-001-a30
4k   11/3      20  tst-nest-002-a00
4k    4/1       5  tst-nest-002-a10
4k    2/1       3  tst-nest-002-a11
4k    1/0       1  tst-nest-002-a12
4k    1/0       1  tst-nest-002-a20
```

Figure 31.4: TAB_CPORT by ctys-vhost

It has to be recognized here, that entries of multiple mounted NFS directories would occur once for each match. The second specific is the handling of multiple NIC interfaces within a VM, which will result in one entry for each NIC.

31.1.2 Append Data to present Database

The following workflow is used to append data for a new processing node to the cacheDB.

1. Make a copy of the raw file database "enum.fdb" to a temporary cacheDB directory, e.g. "$HOME/.ctys/db/tmp". It is also possible to work completly on a copy, than additionally at least the "macmap.fdb" is required.

```
mkdir $HOME/.ctys/db/tmp

cp  \
  $HOME/.ctys/db/default/enum.fdb \
  $HOME/.ctys/db/tmp
```

2. Call the update of the data by "append" for data of session type "QEMU".

```
ctys-vdbgen \
  -t qemu \
  -T all \
  --progress
  --cacheDB=$HOME/.ctys/db/tmp \
  --append \
  --base=qemu  \
  lab00'(-d 2,s:16,w:0,p)'
```

For the curious: The usage of "-T all" makes life somewhat easier here, because QEMU reuiqres the load of it's clients too:"CLI,X11,VNC", which is no longer performed, when a preload is selected at all. Omitting the "-t" and "-T" options just defaults to "-t all -T all", which works fine, but requires longer because it scans for all, thus the same directory multiple times.

3. Check the actual collected data by a simple diff.

```
diff \
  $HOME/.ctys/db/default/enum.fdb \
  $HOME/.ctys/db/tmp/enum.fdb
```

4. Copy the new file database into your working directory.

```
cp  \
  $HOME/.ctys/db/tmp/enum.fdb \
  $HOME/.ctys/db/default
```

5. Call ctys-vhost, any display call on the actual data may detect the time-stamp and performs a rebuild of the 2.stage cacheDB. The following leads to all "VMs" of type "QEMU" on the machine "lab00"

```
ctys-vhost -o l,pm,id QEMU tst lab00
```

When using a macro e.g. the following is displayed.

```
ctys-vhost macro:vconf,sort QEMU tst lab00
```

```
label |stype|TCP           |MAC               |ID
------+-----+------------+-----------------+------------------------------
tst000|QEMU |192.168.1.130|00:50:56:13:11:30|qemu/tst-ctys/tst000/tst000.conf
tst001|QEMU |192.168.1.131|00:50:56:13:11:31|qemu/tst-ctys/tst001/tst001.conf
tst102|QEMU |192.168.1.222|00:50:56:13:11:42|qemu/tst-ctys/tst102/tst102.conf
tst105|QEMU |192.168.1.225|00:50:56:13:11:45|qemu/tst-ctys/tst105/tst105.conf
tst006|QEMU |192.168.1.136|00:50:56:13:11:36|qemu/tst-ctys/tst006/tst006.conf
tst007|QEMU |192.168.1.137|00:50:56:13:11:37|qemu/tst-ctys/tst007/tst007.conf
tst108|QEMU |192.168.1.228|00:50:56:13:11:48|qemu/tst-ctys/tst108/tst108.conf
tst113|QEMU |192.168.1.233|00:50:56:13:11:4D|qemu/tst-ctys/tst113/tst113.conf
tst121|QEMU |192.168.1.205|00:50:56:13:11:55|qemu/tst-ctys/tst121/tst121.conf
tst124|QEMU |192.168.1.208|00:50:56:13:11:58|qemu/tst-ctys/tst124/tst124.conf
tst127|QEMU |192.168.1.211|00:50:56:13:11:5B|qemu/tst-ctys/tst127/tst127.conf
tst145|QEMU |192.168.1.249|00:50:56:13:11:6d|qemu/tst-ctys/tst145/tst145.conf
tst146|QEMU |192.168.1.100|00:50:56:13:11:6e|qemu/tst-ctys/tst146/tst146.conf
tst147|QEMU |192.168.1.101|00:50:56:13:11:6f|qemu/tst-ctys/tst147/tst147.conf
tst148|QEMU |192.168.1.102|00:50:56:13:11:70|qemu/tst-ctys/tst148/tst148.conf
```

The available macro definitions of combined macros only could be displayed with

```
ctys-macros -c -D
```

The processing of the "enum.fdb" only requires frequently some seconds, whereas the(in current version deactivated) groups-cache may take several minutes, because any level of nested includes is valid as an entry point. Though even a small set of nested groups, particularly within stacked environments, could lead to a considerable number of "trees", but this is of course what CACHES are for.

6. That's it.

For resonable scans it should not take more than 1-2 minutes. Could require longer on outdated machines, and/or for deep filesystem structures.

31.1.3 vm.conf Coallocated with Hypervisor

The current version requires for a scan of the GuestOS attributes to be stored within the cacheDB generally the activation of the VM. This is due to the analysis of the actual runtime system. The required parameters could be edited manually too of course, but this might not be the preferred approach. The same requirement of an active GuestOS is given to the scan of available VMs as an upper stack-peer within the GuestOS. When the restrictive CONTEXT check is required, this remains the only applicable approach, beneath the manual edit of the database. The consequence resulting of this is the required start of any valid combination of a VMSTACK to be able to operate by previous pass of pre-checks. This consequently would have to be applied to any permutation of the stack combination to be managed, when restrictive checks are required.

Anyhow, in the majority of practical requirements a restrictive check for specific execution context is not required. Therefore a simple trick results in a dramatic reduction of required instances to be scanned, when the CONTEXT argument of suboption STACKCHECK is applied in order to suppress the locality checks. In this case any machine could be started at any locality and could be initially executed at any valid location once the SACKREQ and HWREQ constraints are applied. This is the default behaviour for any stackable VM capable of execution on upper layers, which is currently QEMU only.

Thus in his case each VM has to be activated only once at an arbitrary location and has to be prepared by "ctys-genmconf" . A copy of the resulting "/etc/ctys.d/vm.conf" file has to be stored within the same directory as the configuration file of the founding hypervisor is. A scan by "ctys-vdbgen" will now detect both, the hypervisor configuration file and the vm.conf of the nested GuestOS without the requirement of changing the runtime state of the VM to ACTIVE. Thus the complete set of data could be scanned offline.

Once this is accomplished, the VM could be started at any location without a new scan. Particularly it's nested upper layer VMs could be executed nested within the GuestOS, whithout an actual specific scan, when prepared based on the same ap-

proach.

It should not be forget, that the synchronity of the cached data shouls not be violated, and is within the responsibility of the user only. No cosistency checks for actual data synchronity are applied.

31.2 LDAP based VM-Stack Nameservices

ffs.

31.3 Group Addressing

31.3.1 Group Caches - The Performance Clue

ffs.

31.3.2 Preconfigured Task-Groups

The usage of preconfigured groups has several advantages, where the first application might be the usage to easily update the cacheDB.

Once a group is configured it could simply be used as a replacement for the <target-application-entity> . The following rebuilds the whole cacheDB for "$USER" by one call, once the group "$USER-all" is configured.

```
ctys-vdbgen $USER-all
```

31.4 Stack Addressing

ffs.

31.5 DHCP Poll

The call of ctys-extractMAClst works on a dhcpd.conf as required for the widely common DHCP daemon. In current version it just extracts static entries for a fixed assignment of

MAC-to-IP-Addresses. Dynamic assigned leases for address pools are not yet supported. So for using two pre-requisites have to be fulfilled:

1. Any participating VM and PM has to have it's own assigned fixed mapping of an DNS-Name, TCP/IP-Address, and MAC-Address.
2. Access to the dhcpd.conf file, e.g. as a copy is required. Another approach might be that for individual users with local caches a prepared macmap.fdb could be provided.

This leads to screen output:

```
ctys-extractMAClst /etc/dhcpd.conf
```

The following call generates the default target " /.ctys/db/default/macmap-fdb" mapping database:

```
ctys-extractMAClst -P /etc/dhcpd.conf
```

A second tool which is called "ctys-extractAPRlst" generates a macmap.fdb too, but it polls the local accessible DHCP server by using ping and evaluates the arp caches. Thus it will work on one segment only, which might be in the era of switched-LANs not a problem. And requires the hosts to be on-line, which is for the "ctys-extractMAClst" tool not required.

This leads to screen output:

```
ctys-extractARPlst host01 host02 group01 host03 group02
```

The following call generates the default target " /.ctys/db/default/macmap-fdb" mapping database:

```
ctys-extractARPlst -P host01 host02 group01 host03 group02
```

When the whole domain has to be scanned, which could include any host, so particularly address-pools too, than the following could be applied:

```
for i in `host -l <mydomain>|awk '{print \$1;}'`;do
  ctys-extractMAClst \$i
done
```

Anyhow, due to the simple format of macmap.fdb, which MS-Excel compatible, this mapping file-db macmap.fdb could be edited manually too.

31.6 PXELinux - Address Translation

One example application for "ctys-vhost" is the resolution of PXE IP entries in the required specific form for PXELinux.

Once the database is setup with "ctys-vdbgen" and "ctys-extractMAClst" or "ctys-extractARPlst" , the generation of PXELinux addresses becomes an action of 0.x seconds, check it out!

The match for the regexpr should be as expected, this has to be considered when choosing the search string. Here it is assumed, that the only host/dns-names matching "inst" are of form "inst00[0-9]", and are foreseen as temporary install addresses for raw-templates, installed by kickstart-files.

The following shell call generates and displays for any inst-machine an suitable PXELinux IP-Address.

Let me mention, that this requires almost only the initial-overhead of 0.3seconds from an database of 120 DHCP entries

when using macmaponly.

When the MAC-Mapping cache is not used, instead the static cache DB based on pre-cached enum.fdb is used, the performance is the initial overhead of 0,5seconds from an database of about 450 VM entries.

Bulk tests will follow, but even when it reaches seconds for about thousand or more entries, it is "almost nothing" compared to a manual search in an distributed environment intermixed with shared mounts and unique local entities.

```
\#!/bin/bash

for i in `ctys-vhost -C macmaponly -o m,d -M all inst`;do
   echo -n "\${i\#*;}=\${i\%;*}->";
   gethostip \${i\#*;};
done"
```

This results to the output:

```
inst000=00:50:56:16:11:00->inst000.soho 192.168.1.80 C0A80150
inst001=00:50:56:16:11:01->inst001.soho 192.168.1.81 C0A80151
inst002=00:50:56:16:11:02->inst002.soho 192.168.1.82 C0A80152
inst003=00:50:56:16:11:03->inst003.soho 192.168.1.83 C0A80153
inst004=00:50:56:16:11:04->inst004.soho 192.168.1.84 C0A80154
inst005=00:50:56:16:11:05->inst005.soho 192.168.1.85 C0A80155
inst006=00:50:56:16:11:06->inst006.soho 192.168.1.86 C0A80156
inst007=00:50:56:16:11:07->inst007.soho 192.168.1.87 C0A80157
inst008=00:50:56:16:11:08->inst008.soho 192.168.1.88 C0A80158
inst009=00:50:56:16:11:09->inst009.soho 192.168.1.89 C0A80159
```

So the MAC address could be set, the alphanumeric IP address could be configured, and the automatic configuration will be performed.

ctys-vhost is more and more excessively used for address translations between arbitrary keys within ctys itself. Beneath it's versatelity it's - at least for small and medium networks - outbursting performance for resolution of various keys is e.g. an almost fixed overhead of about 0.3 to 0.6 mseconds(on a medium sized reference machine), with a few additional processing time for some tasks only.

31.7 Formatted output by Generic Tables

The output of ctys-vhost could be formatted by the same means as for LIST and ENUMERATE of ctys. Therefore the "-o" output flag supports the common "tab_gen" option. The usage of macros is supported in exactly the same way as for ctys.

The following call

```
ctys-vhost -o macro:TAB_CPORT olymp xen acue
```

displays the result:

```
Label |stype|cport|PM         |MAC               |TCP
------+-----+-----+----------+------------------+-------------
tst100|XEN  |     |olymp.soho|00:50:56:13:11:40|192.168.1.220
tst101|XEN  |     |olymp.soho|00:50:56:13:11:41|192.168.1.221
tst104|XEN  |     |olymp.soho|00:50:56:13:11:44|192.168.1.224
```

Figure 31.6: TAB_CPORT by ctys-vhost

The output here is of course similar to the ENUMERATE ("TAB_CPORT by ENUMERATE") content, and is actually a pre-cached output of ENMUERATE.

The complemantary output of dynamic runtime data is presented by LIST action in "TAB_CPORT by LIST" .

The default output when the "-o" option is suppressed is to display a table named as "TAB_CTYS_VHOST_DEFAULT". which produces e.g. for the call

```
ctys-vhost QEMU acue app1
```

The output

```
label   |stype|distro   |distrorel    |os     |osrel    |PM       |TCP
--------+-----+---------+-------------+-------+---------+---------+-------------
tst102  |QEMU |debian   |4.0r3        |Linux  |2.6.16   |app1.soho|192.168.1.222
tst000  |QEMU |CentOS   |5            |Linux  |2.6.18   |app1.soho|192.168.1.130
tst001  |QEMU |CentOS   |5            |Linux  |2.6.18   |app1.soho|192.168.1.131
tst108  |QEMU |debian   |4.0r3        |Linux  |2.6.16   |app1.soho|192.168.1.228
tst105  |QEMU |Fedora   |8            |Linux  |2.6.18   |app1.soho|192.168.1.225
tst104  |QEMU |SuSE     |9.3          |Linux  |2.4      |app1.soho|192.168.1.224
tst113  |QEMU |OpenSuSE |10.3         |Linux  |2.6      |app1.soho|192.168.1.233
tst122  |QEMU |Ubuntu   |6.06.1-LTS-S |Linux  |2.6      |app1.soho|192.168.1.206
tst007  |QEMU |Ubuntu   |7.10         |Linux  |2.6      |app1.soho|192.168.1.137
tst124  |QEMU |OpenBSD  |4.0          |OpenBSD|4.0      |app1.soho|192.168.1.208
tst123  |QEMU |OpenBSD  |4.2          |OpenBSD|4.2      |app1.soho|192.168.1.207
tst127  |QEMU |OpenBSD  |4.3          |OpenBSD|4.3      |app1.soho|192.168.1.211
tst006  |QEMU |W2K      |2000-WS      |Windows|2000-WS  |app1.soho|192.168.1.136
tst121  |QEMU |Ubuntu   |8.04-LTS-D   |Linux  |2.6      |app1.soho|192.168.1.205
tst125  |QEMU |Ubuntu   |8.04-LTS-S   |Linux  |2.6      |app1.soho|192.168.1.209
arm-test|QEMU |QEMU     |0.9.1        |Linux  |2.6.17-r |app1.soho|192.168.1.229
small   |QEMU |QEMU     |0.9.1        |NetBSD |4.0      |app1.soho|192.168.1.229
sparc-1 |QEMU |         |             |Linux  |2.6.11+t |app1.soho|192.168.1.229
sparc-1 |QEMU |         |             |Linux  |2.6.11+t |app1.soho|192.168.1.229
linux   |QEMU |QEMU     |0.9.1        |Linux  |2.6.17-r |app1.soho|192.168.1.229
coldfire|QEMU |QEMU     |0.9.1        |Linux  |2.6.17-r |app1.soho|192.168.1.229
```

Figure 31.7: Default table for ctys-vhost: TAB_CTYS_VHOST_DEFAULT

31.8 Filter by awk-Regular Expressions

31.8.1 Datastreams and Match Conditions

The cacheDB is organized set of lines within a flat file,where each record is a semicolon seperated list of attributes. This is compatible to almost any office spreadsheet program and could be easily imported to any relational DBMS. The actual column

names including their canonical position index could be printed with the TITLEIDX option.

Even though a column structure is present the majority of queries will be performed to the whole line as a flat string pattern, which delivers a fast match.

A repetitive application of multiple selection strings for application of the intermidiate results of the previous is supported. Alternatively some awk boolean operations could be applied.

Each given select string "<selector>" is evaluated by the awk-regexpr "$0 s", where the variable "s" has the value "<selector>". A valid value couls be "." for any character, or it could be "(test104|tst153)" for matching any record which contains either "test104" of "tst153".

Anyhow, some enhancements and field-oriented features are "on the road".

31.8.2 Wildcards and Ambiguity

The arguments of "ctys-vhost" are bypassed tranparently to a awk-regexp and are evaluated each as a chained filter on the results of the previous argument.
The call with regular expressions in general matches on "$0" of awk, thus may lead to some specific results, when not considered thoroughly.

A typical pitfall might be, when the user has for each type of VM it's own subdirectory, with lowercase directory name of the hypervisor. For QEMU namely "qemu". When the following query is executed, this just scans the database and matches any record containing the string "qemu".

```
ctys-vhost -o ids qemu vadmin
```

The result is the subset of IDs(conf-file-paths) for "qemu", where the string "vadmin" matches. The following result is

displayed

```
/homen/vadmin/qemu/tst/arm-test/arm-test.ctys
/homen/vadmin/qemu/tst/small/small.ctys
/homen/vadmin/qemu/tst/sparc-test/sparc-1.ctys
/homen/vadmin/qemu/tst/sparc-test/sparc-2.ctys
/homen/vadmin/qemu/tst/linux/linux.ctys
/homen/vadmin/qemu/tst/coldfire-test-0.1/coldfire-test-0.1.ctys
/homen/vadmin/vmware/develop/build/linux001.i386-4qemu/linux001.vmx
/homen/vadmin/vmware/develop/build/linux001.i386-4qemu.20070207_1/linux001.vmx
```

Figure 31.8: ctys-vhost awk-regexpr-1

which contains the lines

```
...
/homen/vadmin/vmware/develop/build/linux001.i386-4qemu/linux001.vmx
/homen/vadmin/vmware/develop/build/linux001.i386-4qemu.20070207_1/linux001.vmx
```

Figure 31.9: ctys-vhost awk-regexpr-1

This is due to a simple string match, where the production VM for the QEMU sources itself is matched as a "QEMU-VM". To avoid this the following regexpr could be used.

```
ctys-vhost -o ids /qemu/ vadmin
```

This results in:

```
/homen/vadmin/qemu/tst/arm-test/arm-test.ctys
/homen/vadmin/qemu/tst/small/small.ctys
/homen/vadmin/qemu/tst/sparc-test/sparc-1.ctys
/homen/vadmin/qemu/tst/sparc-test/sparc-2.ctys
/homen/vadmin/qemu/tst/linux/linux.ctys
/homen/vadmin/qemu/tst/coldfire-test-0.1/coldfire-test-0.1.ctys
```

Figure 31.10: ctys-vhost awk-regexpr-1

Finally the correct call would be:

```
ctys-vhost -o ids QEMU vadmin
```

The name of the hypervisor, which is the name of the plugin within ctys, is stored in the database literally, thus as uppercase "QEMU".

Anyhow, the usage of this string within a pathname could still lead to unexpected results.

Part V

Appendices

Chapter 32

Current Loaded Plugins

This section enumerates the current loaded static libraries and the dynamic loaded plugins. Which will be partly detected automaticaly and loaded as predefined or On-Demand.

The following list is generated with the call:

"ctys -T all -v"

REMARK: For limited environents this could produce errors due to memory exhaustion. The error messages ar not obvious!!!

```
------------------------------------------------------------------------
PROJECT          = Unified Sessions Manager
------------------------------------------------------------------------
CALLFULLNAME     = Commutate To Your Session
CALLSHORTCUT     = ctys

AUTHOR           = Arno-Can Uestuensoez - acue@UnifiedSessionsManager.org
MAINTAINER       = Arno-Can Uestuensoez - acue_sf1@sourceforge.net
VERSION          = 01_04_001A02
DATE             = 2008.03.17

LOC              = 59778 CodeLines
LOC-BARE         = 32498 BareCodeLines (no comments and empty lines)
LOD              = 14531 DocLines

TARGET_OS        = Linux:CentOS/RHEL - BSD:OpenBSD
TARGET_VM        = VMware, Xen, QEMU
TARGET_WM        = X11, Gnome, fvwm
GUEST_OS         = ANY(some with limited native-acces support)
```

```
-------------------------------------------------------------------------
COPYRIGHT        = Arno-Can Uestuensoez - acue@UnifiedSessionsManage
LICENCE          = GPL3
-------------------------------------------------------------------------
EXECUTING HOST   = ws2.soho
-------------------------------------------------------------------------
LIBRARIES(static-loaded - generic):

  Nr   Library                                          Version
  ------------------------------------------------------------
  00   bootstrap.01.01.001                              01.02.002c01
  01   base                                             01.03.001b01
  02   libManager                                       01.02.002c01
  03   cli                                              01.02.002c01
  04   misc                                             01.02.002c01
  05   security                                         01.02.002c01
  06   help                                             01.02.002a01
  07   geometry                                         01.02.002c01
  08   wmctrlEncapsulation                              01.02.002c01
  09   groups                                           01.01.001a01
  10   network                                          01.01.001a01


PLUGINS(dynamic-loaded - ctys specific):

  Nr   Plugin                                           Version
  ------------------------------------------------------------

  00   CORE/CLI                                         01.02.002c01
  01   CORE/COMMON                                      01.02.002c01
  02   CORE/CONFIG/hook                                 01.02.002c01
  03   CORE/CONFIG/config.Linux                         01.02.002c01
  04   CORE/DIGGER/hook                                 01.02.002c01
  05   CORE/DIGGER/list                                 01.02.001b01
  06   CORE/ENV                                         01.02.002c01
  07   CORE/EXEC                                        01.02.002c02
  08   CORE/GENERIC                                     01.02.002c01
  09   CORE/HELP                                        01.02.002c01
  10   CORE/LABELS                                      01.02.002c01
  11   CORE/STACKER                                     01.02.002c01

  12   GENERIC/hook                                     01.02.001b01
  13   GENERIC/LIST/list                                01.02.001b01
```

```
  14   GENERIC/ENUMERATE/enumerate                01.02.001b01

  15   HOSTs/CLI/hook                             01.01.001a02
  16   HOSTs/CLI/session                          01.01.001a01
  17   HOSTs/CLI/list                             01.01.001a00
  18   HOSTs/CLI/info                             01.02.001b01
  19   HOSTs/VNC/hook                             01.02.001b01
  20   HOSTs/VNC/session                          01.02.001b01
  21   HOSTs/VNC/list                             01.02.001b01
  22   HOSTs/VNC/info                             01.02.001b01
  23   HOSTs/X11/hook                             01.01.001a02
  24   HOSTs/X11/session                          01.01.001a01
  25   HOSTs/X11/list                             01.01.001a00
  26   HOSTs/X11/info                             01.02.001b01

  27   VMs/QEMU/hook                              01.01.001a00pre
  28   VMs/QEMU/config                            01.01.001a01pre
  29   VMs/QEMU/session                           01.01.001a00pre
  30   VMs/QEMU/enumerate                         01.01.001a00pre
  31   VMs/QEMU/list                              01.00.001a00pre
  32   VMs/QEMU/info                              01.01.001a00pre
  33   VMs/VMW/hook                               01.02.001b01
  34   VMs/VMW/session                            01.02.001b01
  35   VMs/VMW/enumerate                          01.02.001b01
  36   VMs/VMW/list                               01.02.001b01
  37   VMs/VMW/info                               01.02.001b01
  38   VMs/XEN/hook                               01.01.001a02
  39   VMs/XEN/config                             01.01.001a01
  40   VMs/XEN/session                            01.01.001a00
  41   VMs/XEN/enumerate                          01.01.001a01
  42   VMs/XEN/list                               01.00.001a00
  43   VMs/XEN/info                               01.01.001a00

  44   PMs/PM/hook                                01.01.001a02
  45   PMs/PM/session                             01.01.001a00
  46   PMs/PM/enumerate                           01.01.001a01
  47   PMs/PM/list                                01.00.001a00
  48   PMs/PM/info                                01.01.001a00


CTYS-INTERNAL-SUBCALLS:

  Nr   Component                                 Version
```

```
  ------------------------------------------------------------
  00   ctys                                          01_04_001A02
  01   ctys-callVncserver                            01_01_002c01
  02   ctys-callVncviewer                            01_02_002c01
  03   ctys-dnsutil                                  01_01_001b01
  04   ctys-extractARPlst                            01_01_001b01
  05   ctys-extractMAClst                            01_01_001b04
  06   ctys-genmconf                                 01_01_001a02
  07   ctys-install                                  01_01_001b04
  08   ctys-install1                                 01_04_001A02
  09   ctys-macmap                                   01_01_001b01
  10   ctys-plugins                                  01_01_001a01pre
  11   ctys-smbutil                                  01_01_001b01
  12   ctys-vdbgen                                   01_01_001a01
  13   ctys-vhost                                    01_02_001a00
  14   ctys-vping                                    01_01_001a01
  15   ctys-wakeup                                   01_01_001a02
  16   ctys-xen-network-bridge                       01_01_001a01

  Tiny-Helpers:

  00   getCurOS                                      OK
  01   getCurOSRel                                   OK
  02   getCurDistribution                            OK
  03   getCurRelease                                 OK
  04   getCurGID                                     OK
  05   pathlist                                      OK


------------------------------------------------------------------------
OPTIONAL/MANDATORY PREREQUISITES:


  bash:GNU bash, version 3.1.17(1)-release (x86_64-redhat-linux-gnu)

  SSH:SSH is on this machine not available

  VNC:VNC Viewer Free Edition 4.1.2 for X - built Mar 14 2007 23:13:5

  wmctrl:wmctrl 1.07
```

```
------------------------------------------------------------------------
CURRENT ARG-MEM-USAGE:

  ArgList(bytes):"env|wc -c" => 2212
```

Chapter 33

File Formats

33.1 Common Tools and Formats

The most common internal file format is the output by MACHINE suboption. This is a semicolon seperated file format, which optionally supports a TITLE line for each field. The format is compatible to MS-Excel.

The most important data collector and generator ctys-vdbgen is just a wrapper for the call of ctys with the action ENUMERATE. The replied data is transparently stored in the main file database "enum.fdb", which could be inspected and edited by any ASC-II editor or a common spreadsheet application.

The caching boosts the performance beginning with the factor of 10, which could be much more, depending of various specific circumstances. Even though it could theoretically slow down, no practical example occurred.

The generated enum.fdb is just a raw static cache of distributed configuration data for VMs, which may lack some additional information. Therefore another file database is generated in order to map TCP/IP data of the Guest-OSs within VMs to their MAC-Addresses and DNS names. This file is called the macmap.fdb.

It could be generated by the usage of the tools ctys-extractMAClst and/or ctys-extractARPlst.

The prefetched databased "'.fdb" are pre-processed on a second level by aggregating and correlating the data into a common final static cache-database "ctysd-vhost.statcache.cdb". Additional dynamic intermediate data my be cached within tmp directory.

The databases could be listed and inspected by the tool ctys-vhost.

ctys-vhost -S list
: Lists source databases of first level cache "fdb", which includes the non-cached group files.

ctys-vhost -C list
: Lists statcache caches of second level prefetch, which includes the cached group files.

Some additional options exist within ctys-vhost for listing of members and various output.

33.2 Groups

The groups concept as described within the introduction supports the assembly of arbitrate entities into a common entity, which will be handled as one logical instance. This includes the nesting of includes of groups.

Therefore a file has to be supported in a defined subdirectory, with the name as the literal group name to be used within ctys arguments as a replacement for a <machine-address>.

The group file supports the keyword "#include" which has to be left-most on a line followed by a groupname. The number of includes and the level of nesting is not limited. Recursion loops are checked and aborted by a pre-defined level.

The subdirectory containing the group will be search from a given list of directories by the path variable "CTYS_GROUPS_PATH", which uses a colon seperated list of directory paths.

The ordinary host entries could be one on each line, or a list of more than one comma-seperated entries on a line. For each

entry could be context specific options set the same as on the command line arguments.

An example with nested includes is supported in the install sub-directory conf/ctys/groups.

Following is an example with context options.

```
#
#This groups contains all machines supporting VMs to be used by
#user acue.
#
#Almost any machine is currently accessible by NFS, thus allows
#a simple means of load-balancing throughout the memgers of
#this group.
#
#The only distinction has to be done by type of VM, which
#depends on the actual running kernel.
#
#
#REMINDER: Any option is dominant from it's first occurance on,
#          until overwritten. This is one of the limits of
#          missing namespaces and smart usable assembled data
#          structures within bash. But anyway the real huge
#          benfit from using bash is the opportunity for almost
#          anyone to add his own modules by simple shell scripts.
#
#          It has to be recognized, that due to some basic
#          requirements - e.g. grouping of potential
#          Desktop/Workspace display, the jobs are re-ordered.

#          So it is recommended, that when any context-option
#          is required, that ALL might be assigned it's own
#          options.
#
#
#          So, it's accepted!
#
#
#OpenBSD PM, for tests only. Supports no VMs, Just OpenBSD-PM and
#common HOSTs-(CLI,X11,VNC)
root@host1'(-a enumerate=machine)'
```

```
#Multi-Display-WS with local emulation for usage of
#Win-Equipment, such as Dymo-LabelWriter 320 with it's own
#drivers within VMWare and W2K.
host2'(-a enumerate=machine)'

#WMware WS workstation, DualOpteron-244
host3'(-a enumerate=machine)' user5@host0'(-a enumerate=machine)'

#Experimental Dual-PIII-Coppermine, yum-install of
#VMware-server/player-variants

host4'(-a enumerate=machine)' user3@host5'(-a enumerate=machine)'

#Main fileserver with DualP-III-Thualatin
host5'(-a enumerate=machine)' user3@host10'(-a enumerate=machine)'

#Backup and Experimental HVM by Core2-Duo-6300
host6'(-a enumerate=machine)' user2@host9'(-a enumerate=machine)'

#Experimantal Celeron - S420
root@host6'(-a enumerate=machine)'

#Experimental Celeron 2,4GHz
root@host7'(-a enumerate=machine)'

#app1 as primary paravirtualized DualOpteron-Xen-Server
host8'(-a enumerate=machine,b:/homen/userA%/home1/xen)'
```

On the ctys call the group could be given context options, even when it's members have context options. But not the include-statements. Due to the philosophy, that the last wins, the outer . this is the CLI options - will replace options left of it, within the groups.

When using ctys-vhost the groups will be cached as a pre-resolved subtree in common database (ENUMERATE) format. This will be recursively done for each group file as described in the following chapters.

33.3 Macros

Thus a macro can contain any part of a call except the command itself. The whole set of required options including the execution target or only a subset of options could be stored within a macro.

The macro and it's content are stored within a file which could be edited by each user or provided as a common defaults file. A macro is defined within the default file "default" which is searched in the order:

1. "$HOME/.ctys/macros/default"
2. "<actual-call-conf-path>/macros/default"

The <actual-call-conf-path> is evaluated from the resolved symbolic link of the call.

The following call syntax is provided:

```
MACRO:(
<macro-name>
[%<macro-file-db>]
  [%(
    ECHO
    |EVAL]
  )
]
)
```

MACROs could be nested and chained as required. Even though the recursion depth could be arbitrary a counter is implemented, which sets a threshold limiting recursive processing. This is set by the configuration variable CTYS_MAXRECURSE. The variable protects all recursion depths, thus should be handled carefully. Default is 15 levels.

The keyword "MACRO" prefixes the actual macro alias with the following parts.

<macro-name>
: The actual name of the alias to be replaced.

<macro-file-db>
The default macro file could be altered by this new filename. The "macros" directories will be scanned for a file with given name.

ECHO
The given macro is inserted by "echo" command into the replacement position, which is the default behaviour.

EVAL
The macro is evaluated on the callers site by "eval" call and the result is inserted into the insertion position.

33.4 Static Import File Databases - fdb

33.4.1 macmap.fdb

This file contains the output of the standard call to one of the tools "ctys-extractMAClst" or "ctys-extractARPlst". Which is a three colon semicolon seperated table, callee file-database. The record format is

<DNS-name>;<MAC-address>;<TCP/IP-address>

The default format is expected by the post-processing tools.

The tools ctys-extractMAClst and ctys-extractARPlst could be used by themself for search and output to stdout too.

ctys-vhost allows by the flag "MACMAP" the optional usage of this database when only output suitable is selected for the "-o" option. Or allows the forced usage by "MACMAPONLY".

ctys-macmap works natively on macmap.fdb.

33.4.2 enum.fdb

The enum.fdb file is the first level cached raw output from the ENUMERATE=MACHINE call. Therefore the call of ctys-vdbgen is used, which is a wrapper for the ctys call. Any option of ctys could be provided for ctys-vdbgen and will be passed through transparently. Therefore the flag "-T" and the given targets, e.g. as a group, could be used to constrain the collected and stored data. Using this features several databases

could be used independently with different scopes of network view.

The record format is as shown with the call "ENUMERATE=MACHINE,TITLE":

```
"ContainingMachine;SessionType;Label;ID;UUID;MAC;TCP/IP;\
VNCAccessPort;VNCBasePort;VNCDisplay;Distribution;OS;\
VersionNr;SerialNr;Category"

host.soho;QEMU;tst100;\
/homen//tst100-01.01.001.x86\_64/tst100-inst.conf;\
;00:50:56:13:11:40;;;;;;;;;
```

Space are not allowed, multiple entries will be ignored, just the first will be used.

33.5 Static Pre-Fetch Cache Databases - cdb

The Pre-Fetch Caches are the second level, where some time consuming preprocessing and correlation is performed. This also includes the finale resolution of the group files by replacing each <machine-address> entry with the appropriate ENUMERATE result, and resolving the whole resulting include-tree.

The common enum.fdb entries are stored within one database file, whereas the groups are resolved with their whole tree for each defined group file.

The record format is the common ENUMERATE format.

33.5.1 grpscache.cdb

The groups cache is basically the same as the enum.fdb, but the entries are for each group the resolved complete include tree.

33.5.2 statcache.cdb

This is the cached enum.fdb. In addition to enum.fdb the macmap.fdb is correlated to any ITCP relevant field. Therefore - if available - any item should now contain a MAC-address and the related TCP/IP-address. This allows by simple search operations the evaluation of the GuestOSs TCP/IP-address from the VMs configuration file. Thus the native access to the GuestOS.

33.6 Dynamic Runtime Cache Databases

Some internal data is cached into files, which could be controlled by flags. This is due to reuse of data spanning more than one call.

33.7 Configuration Entries for ctys

33.7.1 Actual Processing vs. Administrative Display

The first point to mention is the actual functionality supported by the various configuration options.

The native configuration files of the various hypervisors support values in order to actually effect their managed VM. The defined values within ctys for this version partly have to be manually edited in order to display and manage the attributes required for managing the sessions only, but have no effect on the behaviour of the VM. Some tools like ctys-genmconf support the automated generation and upgrade of configuration snapshots.

Even though there is some additional effort to spent, the gain is an integrated database containing the whole directory of available systems and required information for their usage and access management. This becomes quickly obvious, when someone has to deal with more than only half a dozen machines. With the UnifiedSessionsManager it is easily possible to manage thousends of VMs within network based filesystems like NFS, where several nodes could access the same set of machines on different paths. Additionally the capabilities of offline VMs

could be easily managed and accessed, which otherwise would be "hidden" within the deactiveated machine.

Anyhow, the parameters displayed with ENUMERATE option are read out from the native configuration with highest priority, where possible. When this fails or is simply not provided, the manual stored data is used.

33.7.2 Configuration File Variants

The various supported VMs have partly quite different configuration options for their VM files. They vary in the content and syntax or, as for example QEMU, even do not support a configuration file, but an extended set of call options. The majority of VMs supports almost(all?) any configuration file options as call parameter too, thus based on this interface a generic and unified configuration file interface could be defined. This file could be more than a passive data file only, but become an active wrapper, which itself is a executable script. This is applied as first application to the ctys-specific wrapper file for the QEMU plugin, and will be extended to the remaining.

Another common lack is that no(or pure) entries related to the GuestOS are supported within the configuration files, but just some virtual hardware related like the MAC address and the UUID. This is perfectly allright, when just dealing with virtual hardware, but opens some challanges, when the running GuestOS has to be incorporated into a system, requiring native access.

The same occurs, when a common management has to be defined for load distribution or relocation of services on different physical entities, where probably some local access to limited resources is required.

Therefore within the UnifiedSessionsManager a common set of information required by the configuration of a VM and it's contained GuestOS is defined. This is just partly required to be defined when already supported by the VMs configuration file, but completely when lacking any information.

Although the introduction of a new format may cause some - but few - additional effort, the gain in combination with the tools like ctys-vhost and ctys-macmap is more than compensatory.

The following keywords are defined and evaluated by the ctys tools. The keywords are pre-fixed with "#@#" in order to provide the possibility to used them embedded into the specific configuration file when appropriate.

Additionally to the specific configuration files of the provided VMs some other predefined files will be scanned. The basic convention for the naming and location of VMs configuration files is defined as:

1. Any VM has it's own directory, wherein all related files are stored. These are the images and the configuration files. This is commonly pre-required for all VMs.
2. The name of the directory is the same as the VM name, which could be the display name as well as a DomU name. For the directory name some variations could be set, e.g. due to versioning or backups. The tools like ctys-vhost search by default for the first match by alphabetical order only, which eliminates the usage of backup files.
3. The configuration file will be named the same as the containing directory, but with the appropriate post-fix, which is e.g. 'vmx' or 'conf'.
4. Additional files, to be optionally used for ctys information when required are:
   (a) Same filenamepath as the VMs config-file, but with the post-fix 'ctys'.
   (b) The same file stored in the parent directory.
   (c) When common definitions to be used, these finally are searched within the CTYSCONF file.

   The scan for resolution will be performed until a match occurs, thus the first match wins.

The order of search is:

1. <vmx-file>

(a) The native configuration file of the VM. The scanning subsystem first tries to match standard information provided by the native syntax of the VM.
(b) Second the ctys specific entries will be scanned.

2. ${<vmx-file>%.*}.ctys
   The ctys-file coallocated within the same directoy as the plugin specific configuration file.

3. ‘dirname <vmx-file>‘.ctys
   The parent directory of the plugins configuration directory.

4. ${CTYSCONF}
   The predefined ctys configuration file.

33.7.3 Keywords

The following keywords are defined:

#@#ARCH
The supported architecture of the VM. For additional description refer to Section 33.7.7 ‘Virtual Hardware-Platform’ on page 634 .

#@#CTYSRELEASE
The identified of the release, which was used to create the current record, helpful for distributed access.

#@#CATEGORY
The category of this Machine. Currently VM or PM.

#@#CSTRG
A private context string supported for the plugin.

#@#DIST
The distribution name of the GuestOS.

#@#DISTREL
The release of the distribution of the GuestOS.

#@#EXECLOCATION

#@#GATEWAY
The gateway to be used by the GuestOS. This could be registered specific to each interface.

#@#HWCAP
The provided harware capacity.

#@#HWREQ
The required harware capacity.

#@#HYPERREF
The release of the hypervisor used to create the VM.

#@#IP[0-9]*
The IP parameter is tightly correlated with the MAC parameter. For additional description refer to Section 33.7.4 'Interface Keywords' on page 631 .

#@#LABEL
Label of VM, which is used as unique name. Could be e.g. the display name or a DomU name.

#@#MAC[0-9]*
The MAC parameter is tightly correlated with the IP parameter. For additional description refer to Section 33.7.4 'Interface Keywords' on page 631 .

#@#MAGICID-<plugin>
A magicID to for classification of the managing plugin for this type. For additional description refer to Section 33.7.5 'MAGICID' on page 634 .

#@#OS
The name of the OS.

#@#OSREL

The release of the GuestOS.

#@#PLATFORM
The supported virtual HW as execution base for the GuestOS. Advanced support for this option is provided by QEMU. For additional description refer to Section 33.7.7 'Virtual Hardware-Platform' on page 634 .

#@#RELOCCAP
The provided reloaction capacity. For additional description refer to Section 33.7.9 'Execution Location and Relocation' on page 637 .

#@#SERNO
A free serial number for administration by the user. The tools generate the recommended format by utilising the "date" tool with the call:

date +%Y%m%d%H%M%S,

which might be sufficently unique, when seen within it's context.

#@#SESSIONTYPE
The type of session. This field will be evaluated by the plugin itself, passed through to the main dispatcher.

#@#SPORT
The serverport for management access to the hypervisor. This is supported by Xen and QEMU.

#@#STACKCAP
The offered stacking capacity of the VM. For additional description refer to Section 33.7.8 'Stacking Entries' on page 637 .

#@#STACKREQ
The required stacking capacity by the VM. For additional description refer to Section 33.7.8 'Stacking Entries' on page 637 .

#@#USTRG
An arbitrary user string. For now just trusted, means no specific validation is done. Particularly the size should be

lept small, which means some bytes only, as a reminder for the VMs task for example.

#@#UUID
The UUID of the machine.

#@#VCPU
The number of configured virtual CPUs.

For Xen called within Dom0 with the "PM" argument used the actual present CPUs, which is the sum of all CPU cores, is additionally displayed.
<Dom0-cpus>/<total-number-of-cores>

#@#VERSION
The release of the OS, which is the kernel release for Linux.

#@#VMSTATE
The state of the VM. For additional description refer to Section 33.7.6 ‘VMSTATE’ on page 634 .

#@#VNCACCESSPORT
The port number of VNC access for the GuestOS, if required to be defined explicitly.

#@#VNCACCESSDISPLAY
The DISPLAY when required to be fixed.

#@#VNCBASEPORT
The baseport when to be calculated with usage of display.

#@#VRAM
The configured RAM for the GuestOS, when called with "VM".

For Xen called within Dom0 with the "PM" argument used the total amount of physical RAM is additionally displayed.
<Dom0-RAM>/<total-physical-RAM>

Additional data could be provided for and by the specific plugin. This is required for example in the specific configuration

file defined for the QEMU plugin by ctys. Which has to be wrapped due to it's call-option only interface, to be unified in accordance to the common usage with the remaining VMs.

33.7.4 Interface Keywords

Specific consideration is required for the interface configuration and it's representation within the GuestOS, particularly when multiple interfaces are configured.

As mentioned before, the first attempt of the ENUMERATE action is to readout the native values of the VM. This is commonly for the MAC address only, where in case of XEN "arbitrary" default values are set when values are not present. In case of VMware the hypervisor eventually resets these values dynamically if not configured to be static.

Thus the first requirement of the UnifiedSessionsManager is the static configuration of MAC addresses.

The next point is the missing interface index within Xen and the varying prefix of the interfacenames within OpenBSD guests for example. Therefore ctys numbers the interface by the order they were found. This is done for the native configured interfaces within the configuration files, where no index is provided. In case of names containing a numbering part, this number is kept. When redundancy occurs, the first match wins, and a warning is displayed for the second, which is dropped. When applicable the binding of interface names to the actualy HW devices should be fixed, this is particularly true, when hotplug Ethernet devices are used. This could be done by configuring MAC addresses into ifcfg-scripts and/or by usage of such tools as "ifrename".

The next step is to evaluate eventually configured TCP/IP parameters, like the IP address from the configuration files. This is currently "somehow" supported by Xen only, but is not indexed too. Thus ctys adds an "IP[0-9*]" options, which provides the whole set of required parameters for the VMs as depicted within the following description. When redundancy occurs, the behaviour is the same as for MAC addresses.

The correlation between the MAC address representing the harware item, and the TCP/IP parameters representing the upper communications protocol layers is given by the index values evaluated before. Thus the first TCP/IP entity with the address "0" is assigned to the first AMC address with the same index. The rest is worked out as might be expected. For TCP/IP addresses without an assigne MAC interface a warning is displayed.

The following syntax is available to be applied.

```
<INDEXEDENTRY>[0-9]* = <entity>

  <entity> =:
    <elements>[,<entity>]

  <elements> =:
    <entry0>[%<entry1>[%<entry2>[%<entry3>[...]]]]
```

The specific adaption for MAC addresses:

```
MAC[0-9]* = <mac-entity>
```

Where a missing index is equal to 0.

The specific adaption for IP:

```
IP[0-9]* = <ip-entity>

<ipentity> =:
  <ip-elements>[,<ip-entity>]

<ip-elements> =:
  [<dotted-IP-addr>]%[<netmask>]%[<relay>]\
  %[<ifname>]%[<ssh-port>]%[<gateway>]
```

Where a missing index is equal to 0.

The values mostly are edited independent from the actual configuration. Thus particularly do not neccesarily represent a static configuration. In case of DHCP these should be configured too, but DHCP might use fixed address assignments.

This is also the style the ctys-extractMAClst utility relies on.

<dotted-IP-addr>
TCP/IP address in numerical form should be preferred. Netmask - if present - has to be provided within it's own field.

<netmask>
The netmask.

<relay> This is the local interconnection device, which could be a virtual bridge, switch/hub or a router. This version supports bridges and switches/hubs only. Support for shorewall will follow soon, check it out.

<ifname>
The name of the interface. Multiple IP addresses on the same interface are supported, thus could have the form "<ifname>:ifnum". The consistency of the interface names is within the responsibility of the user.

<ssh-port>
An alternate port for SSH. OpenSSH supports different ports for each interface.

<gateway>
An individual gateway for networks to be accessed by current interface. Additional settings are required, else the default is used.

The following is an example for configuration of the IP addresses only.

```
#@#IP2 = "11.0.0.11,11.0.11.0,11.11.0.0"
```

The next example provides the full scope of possible information to be stored.

```
#@#IP3 = \
  "11.0.0.1%255.255.255.0%xenbr0%eth3%222,\
   11.0.1.0%255.255.255.0%xenbr1%eth3:1%223"
```

Yes, the <CR> is for LaTeX only.

33.7.5 MAGICID

The MAGICID defines specific behaviour for the filesystem scanners how to recognize the found configuration. When the plugin value only is found, it will immediately accept the file as a valid configuration.

The values could be used for <plugin> in order to control the scan behaviour of ENUMERATE.

- <plugin>
  The SESSIONTYPE of the owning plugin.
- IGNORE
  Ignores the file without any further processing.
- NOENUM
  ffs.

33.7.6 VMSTATE

The state of the VM, currently ACTIVE is supported only.

33.7.7 Virtual Hardware-Platform

The virtual hardware platform is closely correlated to the architecture supported by the VM. These are dependent on the type and capabilities of the utilized hypervisor.

Currently the following non-comprehensive list of supported architectures and platforms is available, which depends partly on the actually used physical hardware base. For additional information refer to the specific documentation of the VMs.

- Architecture - ARCH
  - i386
    VMWARE, XEN, QEMU
  - x86_64, amd64
    VMWARE, XEN, QEMU
  - arm9
    QEMU, for an example with debian installation refer to Section 22.1.2 'Debian-4.0_r3 for ARM' on page 453 , for the provided example by QEMU refer to Section 22.1.9 'Qemu Test and Demo Examples' on page 463 .
  - coldfire, PowerPC, MIPS, SPARC, SH3, ...
    For display of a list of the current installed release of QEMU the INFO action could be used. An example display is provided within Section 22.1.8 'INFO' on page 460 .

    Preconfigured VM configuration files requireing less adaption to local filesystem are available for the provided examples by QEMU, for additional information refer to .

- Platform - PLATFORM
  - pc-standard-platform
    A standard PC platform with widely available emulated hardware components is supported by VMware, Xen, and QEMU. This includes particularly the option to configure the available RAM and the number of present CPUs.
  - ...
    QEMU supports a variety of architectures of standard

and embedded devices. For additional information refer to user-manual.

The hardware relevant entries describe the present hardware. Therefore the following generic format to each entry is applied.

```
<entity> =:
[<#cnt>x]<elements>[%<add-elements>][,<entity>]

<elements> =:
<component>-<version>-<architecture>[-<opt-attr>]
<opt-attr> =: <attr>[-<opt-attr>]
<#cnt>     =: number present

<add-elements> =: <elements>[%<add-elements>]
```

The overall order of the entries contain the following elements, which could be either physical or virtual.

1. CPUs:
   ```
   <vendor>-<family>-<model>-<stepping>-<frequency>-<cache>\
   [-<VM-cap>]

   <VM-cap> := (SVM|VMX)|PAE
   ```

   This format could vary for non-Linux OSs. Non applicable or available but mandatory fields are padded with dots('.').

2. RAM:
   ```
   <RAM>,<SWAP>
   ```

3. HDD:
   ```
   <device-lst> := <device>[%<device-lst>]
   ```

4. FS:

```
<home-lst> := <home>[0-9]*-<size>[%<home-lst>]
```

Currently the home-partitions onyl are displayed by "ctys-genmconf".

33.7.8 Stacking Entries

The stack relevant entries describe the offered VM and PM capacity as well as the required base. Therefore the following generic format to each entry is applied.

```
<entity> =: <elements>[%<add-elements>][,<entity>]

<elements> =:
<component>-<version>-<architecture>[-<opt-attr>]

<opt-attr> =: <attr>[-<opt-attr>]

<add-elements> =: <elements>[%<add-elements>]
```

The overall order of the entries contain the following elements.

1. VM plugin.
2. A list of present hypervisor capabilites.
3. Release of present stack support.
4. Additional information.

33.7.9 Execution Location and Relocation

The execution locations and the supported and/or allowed relocation behaviour. Currently just for display. For current supported values refer to RELOCCAP and EXECLOCATION .

Chapter 34

LDAP Formats

Chapter 35

Call Input-Output Formats

The following data is required by the subdispatcher in order to integrated data from executed plugin specific functions. The data is expected to be piped to stdio with the given format.

35.1 Common Data Import/Export Options

Due to the various suboptions of the user's call, the internal call is normalized by the flag MACHINE. In some old applications still an extra TERSE flag is required in order to generate pure machine format for postprocessing.

Any other suboption for user output may be provided as defined for the top-most call interface.

35.2 ENUMERATE

35.3 LIST

35.4 INFO

35.5 SHOW

Chapter 36

Miscellaneous

36.1 Basic EXEC principle

The basic execution principle of ctys is first to analyse the given options and build up an call array. Therefore several distinctions have to be made as resulting from permutation and superposing of the expanded CLI arguments. These array is finally executed in sets dependent from multiple criteria. Some examples are:

- Grouping of common sessions for each desktop, due to reliability and addressing gaps when shifting windows between desktops.
- Grouping of sessions for each remote server, but only if not ConnectionForward is choosen, because the current OpenSSH release does not support MuxDemux of multiple XSessions with different Displays.
- Splitting of Remote Server and Local Client execution of ctys, for VMs even though the VM-configuration is available on the server site only, which requires a remote component of ctys to be executed.
- ...and so on.

The implementation of ctys is pure bash with usage of the common shell components such as awk and sed. The whole design is based on unique set of sources which will be executed as the local initial call and client starter part as well as the remote server execution script. In case of DISPLAYFORWARDING the local component just initiates the remote co-allocated execution of Client and Server component. For the case of LOCALONLY

both will be locally executed, thus the ctys acts locally as initial caller, server starter and client starter script.

To assure consistency and compatibility the remote and local versions will be checked and the execution is proceeded only in case of an match of both versions.

36.2 PATH

First of all:

> This is normally just required when handling different versions during development.

Due to the several execution parts locally and remotely located, some extension of PATH is frequently required.

This is particularly true, when during development a version is executed which is not contained within the standard PATH. This is particularly to be recognized, when executing the remote component which relies on the PATH mechanism too.

Due to compatibility issues diverging versions will be rejected, which could be controlled by "-F" option but should be avoided.

Therefore the two environment variables are defined:

> R_PATH: Replaces path for remote execution. L_PATH: Replaces path for local execution.

The local component L_PATH is required for local execution too, because the following subcalls of ctys will be executed based on PATH mechanism, which is most often different to initial path-prefixed test-call.

For example this is for calling a test version for starting a local client and remote server from a test path without changing PATH:

```
V=01\_01\_007a01;\
export R\_PATH=/mntbase/ctys/src/ctys.\$V/bin:\$PATH;\
export L\_PATH=\$R\_PATH;\
```

```
ctys.\$V/bin/ctys -t vmw \
   -a create=b:\$HOME/vmware/dir2\%\$HOME/vmware/dir3,\
      l:"GRP01-openbsd-4.0-001",REUSE
   -g 800x400+100+300:3 \
   -L CF \
   -- '(-d 99)' app2
```

The same for common user with standard install will be:

```
ctys -t vmw \
     -a create=\
        base:\$HOME/vmware/dir2\%\$HOME/vmware/dir3\
         ,label:"GRP01-openbsd-4.0-001"\
         ,REUSE
     -g 800x400+100+300:3 \
     -L CF \
     app2
```

36.3 Configuration files

The configuration of ctys is performed in 4 steps, first has highest priority.

1. Environment Variable If an environment variable is set, it dominates other settings and it's value is kept.
2. $HOME/.ctys/ctys.conf Config-File sourced: $HOME/.ctys/ctys.conf
3. <install dir>/conf/ctys.conf Config-File sourced: <install dir>/conf/ctys.conf
4. Embedded defaults in ctys.

36.4 Media Access Control(MAC) Addresses - VM-NICs

This is just an short extract of repetition for understanding WHY a VMs MAC-address should begin with either '2', or '6', or 'A', or 'E' - shortly [26AE].

The knowledge of this is an mandatory and essential building block, when assigning addresses to NICs - a.k.a. VNICs - of VMs for participation of the VM on LAN communications. So will be given thoroughly here.

First of all - this item is described excellently in the book of Charles E. Spurgeon[45] at pg. 42.

Application hints with general visual VM-Networking explanation and a short sum-up for application of MAC-Addresses on VMs are available at the Xen-Wiki[114].

The standards are available at ieee.org[140].

Now the details. The basis for this numbering are the so called DIX and IEEE 802.3 standards. The following items give the explanation, even though anyone should draw a blueprint of byte-bit-construction once by himself:

- Multicast-bit - by DIX and IEEE 802.3
  The Ethernet frames use the first bit of destination address for distinction between:
  - an explicitly addressed single target - a.k.a. physical or unicast address.
  - a group of recipients with an logical address - a.k.a. multicast address

  The syntax is given by most significant bit in Network Order:

    0: unicast
    1: multicast

  Which is 'X' for frames bit-stream:

    Xnnn mmmm rrrr ssss ....
- Locally and Globally Administered Addresses - IEEE 802.3
  This is defined for IEEE 802.3 only. This bit defines the namespace of (to be cared of!) unambiguity for the given address due to it's administrators area of responsibility.
  - globally administered addresses To be used by public - so globally coordinated - access, which has to prevent anyone from buying two NICs with the same MAC-Address.
  - locally administered addresses Could be used according to policies of locally responsible administrators.

This is particularly required for management of VMs, when these should be used in bridged mode, which is a transparently complete host network access as for any physical host.

The syntax is given by second significant bit in Network Order:

0: globally administered
1: locally administered

Which is 'Y' for frames bit-stream:

nYnn mmmm rrrr ssss ....

- Distinction of Network-Order and Representation-Order So far so good, but when trying to define the concrete addresses now, some little details has to be recognized first.

  The given control bits from the network standards are related to networking - of course - thus address positions in network streams as bit-representation.

  But the MAC-Address - 48bit - are written as 6 Octets of hexadecimal nibbles seperated by colons - for human readability.

  The difference of both for the actual "bit-order" arises from the "different logical handling units" for the actual set of bits.

  Whereas the Network-Order assumes a bit as unit, the Representation Order assumes nibbles grouped to octets as handling units.

  So the definition of both units are:
  - Network-Order:
    bit as unit, and a constant bit-stream indexed incrementally beginning with the first bit
  - Representation-Order:
    nibble as unit, grouped to octets as least-significant nibble - containing the least significant bits of a bit stream - first

  Thus the resulting mapping is given by:

  }
  - Network-Order:

```
  nnnn mmmm rrrr ssss ....
  N    M    R    S   ....

- Representation-Order, where additionally the
  bit-order within the nibble is swapped by definition:

  MN:SR:...

  with N from n0-n1-n2-n3 to N3-N2-N1-N0.

- Network:          0001 = 0x1
- Representation:   1000 = 0xF
```

Assuming that for a VM only addresses of following types should be used or to say 'are valid':

```
unicast + locally administered

OK, now after all this results to :

  01nn mmmm rrrr ssss ...

...which is represented as:

  M{nn10}:SR:...

...so has even values only beginning with 2 -  N=2+n*4:

  {nn10}={2,6,10,14}={0x2,0x6,0xA,0xE}=[26AE]

...finally referring to the guide on "XenNetworking-Wiki":

  "aA:..." is a valid address, whereas "aB:...." is not.
```

Mentioning this for completeness - any value of a MAC-Address, where the second nibble of the leftmost octet has one of the values [26AE], is valid.

So, ...yes, no rule without exception.

When dealing with commercial products, free or not, any addressing-pattern could be predefined for manual and generic

MAC-Address assignment within a valid "private" range of the products supplier. This is the case when the first 3 octets of the MAC-Address are defined to be fixed - which is e.g. the suppliers globally assigned prefix - whereas any numbering range could be defined within the following 3 octets.

The given convention should be recognized, because it might be checked by any undisclosed hardcoded piece of code.

For details refer to the specific manuals when required.

"ctys" supports the display of MAC-Addresses as it does UUIDs by action ENUMERATE. This could be used to check uniqueness and might be supported as a ready-to-use MACRO.

Chapter 37

GNU GENERAL PUBLIC LICENSE - Version 3

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you h certain responsibilities if you distribute copies of the software, or you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, rece or can get the source code. And you must show them these terms so th know their rights.

Developers that use the GNU GPL protect your rights with two steps (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly expla that there is no warranty for this free software. For both users' an authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufactu can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for th products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versi of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures th patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an offici standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, oth than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential componen (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means al the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available fr programs which are used unmodified in performing those activities bu which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of

copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice;

keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the cod
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey
and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications t
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditi

a) The work must carry prominent notices stating that you modifi
it, and giving a relevant date.

b) The work must carry prominent notices stating that it is
released under this License and any conditions added under secti
7. This requirement modifies the requirement in section 4 to
"keep intact all notices".

c) You must license the entire work, as a whole, under this
License to anyone who comes into possession of a copy. This
License will therefore apply, along with any applicable section
additional terms, to the whole of the work, and all its parts,
regardless of how they are packaged. This License gives no
permission to license the work in any other way, but it does not
invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must displa
Appropriate Legal Notices; however, if the Program has interacti
interfaces that do not display Appropriate Legal Notices, your
work need not make them do so.

A compilation of a covered work with other separate and independen
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit. Inclusion of a covered wor
in an aggregate does not cause this License to apply to the other

parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is

available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provi
you inform other peers where the object code and Corresponding
Source of the work are being offered to the general public at no
charge under subsection 6d.

A separable portion of the object code, whose source code is exclu
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means a
tangible personal property which is normally used for personal, fami
or household purposes, or (2) anything designed or sold for incorpor
into a dwelling. In determining whether a product is a consumer pro
doubtful cases shall be resolved in favor of coverage. For a partic
product received by a particular user, "normally used" refers to a
typical or common use of that class of product, regardless of the st
of the particular user or of the way in which the particular user
actually uses, or expects or is expected to use, the product. A pro
is a consumer product regardless of whether the product has substant
commercial, industrial or non-consumer uses, unless such uses repres
the only significant mode of use of the product.

"Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to ins
and execute modified versions of a covered work in that User Product
a modified version of its Corresponding Source. The information mus
suffice to ensure that the continued functioning of the modified obj
code is in no case prevented or interfered with solely because
modification has been made.

If you convey an object code work under this section in, or with,
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of th
User Product is transferred to the recipient in perpetuity or for a
fixed term (regardless of how the transaction is characterized), the
Corresponding Source conveyed under this section must be accompanied
by the Installation Information. But this requirement does not appl
if neither you nor any third party retains the ability to install
modified object code on the User Product (for example, the work has
been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material,
requiring that modified versions of such material be marked in
reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors
authors of the material; or

e) Declining to grant rights under trademark law for use of some
trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that
material by anyone who conveys the material (or modified versions
it) with contractual assumptions of liability to the recipient, f
any liability that these contractual assumptions directly impose
those licensors and authors.

All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10. If the Program as yo
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term. If a license document contain
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in th
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly
provided under this License. Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever

licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone

to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not

excuse you from the conditions of this License. If you cannot convey
covered work so as to satisfy simultaneously your obligations under t
License and any other pertinent obligations, then as a consequence yo
not convey it at all. For example, if you agree to terms that obliga
to collect a royalty for further conveying from those to whom you con
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU Affero General Public License into a singl
combined work, and to convey the resulting work. The terms of this
License will continue to apply to the part which is the covered work
but the special requirements of the GNU Affero General Public License
section 13, concerning interaction through a network will apply to th
combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new version
the GNU General Public License from time to time. Such new versions
be similar in spirit to the present version, but may differ in detail
address new problems or concerns.

Each version is given a distinguishing version number. If the
Program specifies that a certain numbered version of the GNU General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation. If the Program does not specify a version number of the
GNU General Public License, you may choose any version ever published
by the Free Software Foundation.

If the Program specifies that a proxy can decide which future
versions of the GNU General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes yo
to choose that version for the Program.

Later license versions may give you additional or different
permissions. However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a

later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest

to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found

    <one line to give the program's name and a brief idea of what it
    Copyright (C) <year> <name of author>

    This program is free software: you can redistribute it and/or mod
    it under the terms of the GNU General Public License as publishe
    the Free Software Foundation, either version 3 of the License, o
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public Licens
    along with this program. If not, see <http://www.gnu.org/licens

Also add information on how to contact you by electronic and paper ma

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program> Copyright (C) <year> <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type
    This is free software, and you are welcome to redistribute it
    under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appro
parts of the General Public License. Of course, your program's comma
might be different; for a GUI interface, you would use an "about box"

  You should also get your employer (if you work as a programmer) or
if any, to sign a "copyright disclaimer" for the program, if necessar
For more information on this, and how to apply and follow the GNU GPL
<http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your p
into proprietary programs. If your program is a subroutine library,
may consider it more useful to permit linking proprietary applicatio
the library. If this is what you want to do, use the GNU Lesser Gene

Public License instead of this License. But first, please read
<http://www.gnu.org/philosophy/why-not-lgpl.html>.

Bibliography

Books

UNIX

[1] Juergen Gulbins: *UNIX Version 7, bis System V.3,* Springer-Verlag Berlin Heidelberg; 1988; ISBN: 3-540-19248-4

[2] Maurice J. Bach: *The Design Og The UNIX Operating System,* Prentice Hall, Inc.; 1986; ISBN: 0-13-201757-1

[3] Sebastian Hetze, Dirk Hohndel, Martin Mueller, Olaf Kirch: *LINUX Anwender Handbuch,* LunetIX Softair; 1994; ISBN: 3-929764-03-2

[4] Rob Flickenger: *LINUX SERVER HACKS,* O'Reilly&Associates, Inc.; 2003; ISBN: 0-596-00461-3

[5] Olaf Kirch: *LINUX Network Administrator's Guide,* O'Reilly&Associates, Inc.; 1995; ISBN: 0-56592-087-2

[6] Daniel P. Bovet, Marco Cesati: *Understanding the LINUX KERNEL,* O'Reilly&Associates, Inc.; 2003; ISBN: 0-596-00002-2

[7] Daniel P. Bovet, Marco Cesati: *Understanding the LINUX KERNEL(2nd. Ed.),* O'Reilly&Associates, Inc.; 2003; ISBN: 0-596-00213-0

[8] Wolfgang Mauerer: *Linux Kernelarchitektur - Konzepte, Strukturen und Algorithmen von Kernel 2.6,* Carl Hanser Verlag Muenchen-Wien; 2004; ISBN: 3-446-22566-8

[9] Alessandro Rubini: *LINUX Device Drivers,* O'Reilly&Associates, Inc.; 1998; ISBN: 1-565592-292-1

[10] Alessandro Rubini, Jonathan Corbet: *LINUX Device Drivers(2nd. Ed.),* O'Reilly&Associates, Inc.; 2001; ISBN: 0-596-00008-1

[11] Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman: *LINUX Device Drivers(3nd. Ed.),* O'Reilly&Associates, Inc.; 2005; ISBN: 0-596-00590-3

[12] Juergen Quade, Eve-Katharina Kunst: *Linux-Treiber entwickeln,* dpunkt.verlag GmbH; 2004; ISBN: 3-89864-238-0

[13] Wehrle, Paehlke, Ritter, Mueller, Bechler: *Linux Netzwerkarchitektur - Design und Implementerung von Netzwerkprotokollen im Linux-Kern,* Addison Wesley, Inc.; 2002; ISBN: 3-8273-1509-3

[14] Brandon Palmer, Jose Nazario: *Secure Architectures with OpenBSD,* Addison Wesley, Inc.; 2004; ISBN: 0-321-19366-0

[15] Michael W. Lucas: *Absolute OpenBSD - UNIX for the Practical Paranoid,* No Starch Press, Inc.; 2003; ISBN: 1-886411-99-9

[16] Chris Tyler: *FEDORA LINUX,* O'Reilly&Associates, Inc.; 2006; ISBN: 1-596-52682-2

[17] Benjamin Mako Hill, Jono Bacon, Corey Burger, Jonathan Jesse, Ivan Krstic: *The Official ubuntu Book,* Pearson Education, Inc.; 2007; ISBN: 0-13-243594-2

[18] Jonathan Oxer, Kyle Rankin, Bill Childers: *UBUNTU Hacks - Tips&Tools for Exploring, Using, and Tuning Linux,* O'Reilly&Associates, Inc.; 2006; ISBN: 1-596-52720-9

[19] Tim SchÃ¼rmann: *(K)Ubuntu - Installieren - Einrichten - Anwenden,* Open Source Press; 2006; ISBN: 3-937514-30-9

[20] Michael Urban, Brian Thiemann: *FreeBSD 6 Unleashed,* Sams Publishing, Inc.; 2006; ISBN: 0-672-32875-5

[21] Marshall Kirk McKusick, George V. Neville-Neil: *The Design and Implementation of the FreeBSD 6 Operating System,* Addison Wesley, Inc.; 2005; ISBN: 0-201-70245-2

Security

[22] Guenter Schaefer: *Netzsicherheit - Algorithmische Grundlagen und Protokolle,* dpunkt.verlag GmbH; 2003; ISBN: 3-89864-212-7

[23] Alexande Geschonneck: *Computer Forensik - Systemeinbrueche erkennen, ermitteln, aufklaeren,* dpunkt.verlag GmbH; 2004; ISBN: 3-89864-253-4

[24] Jonathan Hassel: *RADIUS - Securing Public Access to Privaze Resources,* O'Reilly&Associates, Inc.; 2002; ISBN: 0-596-00322-6

[25] Jason Garman: *Kerberos - The Definitive Guide,* O'Reilly&Associates, Inc.; 2003; ISBN: 1-596-00403-6

[26] Daniel J. Barret, Richard E. Silverman & Robert G.Byrnes: *SSH The Secure Shell - The Definitive Guide,* O'Reilly&Associates, Inc.; 2005; ISBN: 1-596-00895-3

[27] John Viega, Matt Messier & Pravir Chandra: *Network Security with OpenSSL,* O'Reilly&Associates, Inc.; 2002; ISBN: 0-596-00270-X

[28] Jonathan Hassel: *RADIUS,* O'Reilly&Associates, Inc.; 2002; ISBN: 0-596-00322-6

[29] Gerald Carter: *LDAP - System Administration,* O'Reilly&Associates, Inc.; 2003; ISBN: 1-596592-491-6

[30] Matthias Reinwarth, Klaus Schmidt: *Verzeichnisdienste - Telekommunikation Aktuell,* VDE Verlag GmbH; 1999; ISBN: 3-8007-2373-5

[31] Dieter Kluenter, Jochen Laser: *LDAP verstehen, OpenLDAP einsetzen,* dpunkt.verlag GmbH; 2003; ISBN: 3-89864-217-8

[32] Daniel J. Barret, Richard E. Silverman & Robert G.Byrnes: *LINUX Security Cookbook,* O'Reilly&Associates, Inc.; 2003; ISBN: 1-565-00391-9

[33] Charlie Scott, Paul Wolfe & Mike Erwin: *Virtual Private Networks,* O'Reilly&Associates, Inc.; 1999; ISBN: 1-565592-529-7

[34] Bill McCarty: *SELINUX - NSA's Open Source Security Enhanced Linux,* O'Reilly&Associates, Inc.; 2004; ISBN: 1-565-007161-7

[35] Rober L. Ziegler: *Linux Firewalls,* New Riders Publishing, Inc.; 2000; ISBN: 0-7357-0900-9

[36] D. Brent Chapman and Elisabeth D. Zwicky: *Einrichten von Internet Firewalls,* O'Reilly&Associates, Inc.; 1996; ISBN: 3-930673-31-2

[37] Wolfgang Barth: *Das Firewall-Buch,* Nicolaus Millin Verlag GmbH; 2004; ISBN: 3-89990-128-2

[38] Joseph Kong: *Designing BSD Rootkits - An Introduction to Kernel Hacking,* No Starch Press, Inc.; 2007; ISBN: 978-1-593271-142-8

[39] Cyrus Peikari, Anton Chuvakin (Peter Klicman, Andreas Bildstein, Gerald Richter): *Kenne deinen Feind - Fortgeschrittene Sicherheitstechniken,* O'Reilly&Associates, Inc.; 2004; ISBN: 3-89721-376-1

[40] Ryan Russel et al.: *Die mitp-Hacker-Bibel,* mitp-Verlag/Bonn; 2002; ISBN: 3-8266-0826-3

[41] Wallace Wang: *Steal This Computer Book 4.0 - What They Won't Tell You About The Internet,* No Starch Press, Inc.; 2007; ISBN: 1-59327-105-0

[42] Johnny Long: *Google Hacking - For Penetration Testers,* Syngress Publishing, Inc.; 2005; ISBN: 1-931836-36-1

[43] Thomas Bechtold, Peter Heinlein: *Snort, Acid & Co. - Einbruchserkennung mit Linux,* Open Source Press; 2004; ISBN: 3-937514-01-3

[44] Syngress Autorenteam: *Snort 2.0 Intrusion Detection,* mitp-Verlag/Bonn; 2003; ISBN: 3-6266-1304-X

Networks

[45] Charles E. Spurgeon: *Ethernet - The Definitive Guide,* O'Reilly&Associates, Inc.; 2000; ISBN: 1-56592-660-9

[46] Olaf Kirch: *LINUX - Network Administrators Guide,* O'Reilly&Associates, Inc.; 1995; ISBN: 1-56592-087-2

[47] Hal Stern, Mike Eisler & Ricardo Labiaga: *Managing NFS and NIS,* O'Reilly&Associates, Inc.; 2001; ISBN: 1-56592-510-6

[48] Paul Albitz & Cricket Liu: *DNS and Bind,* O'Reilly&Associates, Inc.; 2001; ISBN: 1-596-00158-4

[49] James M. Kretchmar: *Open Source Network Administration,* Pearson Education, Inc.; 2004; ISBN: 0-13-046210-1

[50] Douglas R. Mauro, Kevin J. Schmidt: *Essential SNMP(1.nd Ed.,* O'Reilly&Associates, Inc.; 2001; ISBN: 0-596-00020-0

[51] Douglas R. Mauro, Kevin J. Schmidt: *Essential SNMP(2.nd Ed.,* O'Reilly&Associates, Inc.; 2005; ISBN: 0-596-00840-6

[52] James D. Murray: *Wimdows NT SNMP,* O'Reilly&Associates, Inc.; 1998; ISBN: 1-56592-338-3

[53] Marshall T. Rose: *The Simple Book(2nd. Ed.),* Prentice Hall, Inc.; 1996; ISBN: 0-13-451659-1

[54] David Perkins, Evan McGinnis: *Understanding SNMP MIBs,* Prentice Hall, Inc.; 1997; ISBN: 0-13-437708-7

[55] Mathias Hein, David Griffiths: *SNMP Simple Network Management Protocol Version 2,* International Thomson Publishing GmbH; 1994; ISBN: 3-929821-51-6

[56] Peter Erik Mellquist: *SNMP++ An Object-Oriented Approach to Developing Network Management Applications,* Hewlett-Packard(TM) Professional Books; 1997; ISBN: 0-13-264607-2

[57] Marshall T. Rose: *The Open Book - A Practical Perspective on OSI,* Prentice Hall, Inc.; 1990; ISBN: 0-13-643016-3

[58] Uyless Black: *Network Management Standards(2nd.Ed.),* McGraw-Hill, Inc.; 1994; ISBN: 0-07-005570-X

[59] Wolfgang Barth: *NAGIOS - System and Network Monitoring,* No Starch Press, Inc.; 2006; ISBN: 1-59327-070-4

[60] Heinz-Gerd Hegering, Sebastian Abeck, Bernhard Neumair: *Integriertes Management vernetzter Systeme,* dpunkt.verlag; 1999; ISBN: 3-932588-16-9

[61] Walter Gora, Reinhard Speyerer: *ASN.1 Abstract Syntax Notation One(2nd.Ed.),* DATACOM-Verlag; 1990; ISBN: 3-89238-023-6

[62] Klaus H. Stoettinger: *Das OSI-Referenzmodell,* DATACOM-Verlag; 1989; ISBN: 3-89238-021-X

Embedded Systems

[63] Derek J. Hatley,Imtiaz A. Pirbhai: *Strategien fuer die Echtzeit-Programmierung,* Carl Hanser Verlag Muenchen-Wien; 1988; ISBN: 3-446-16288-7

[64] Edmund Jordan: *Embedded Systeme mit Linux programmieren - GNU-Softwaretools zur Programmierung ARM-basierender Systeme,* Franzis Verlag GmbH; 2004; ISBN: 3-7723-5599-4

[65] Michael Barr, Anthony Massa: *Programming Embedded Systems(2nd.Ed.),* O'Reilly&Associates, Inc.; 2006; ISBN: 1-596-00983-6

[66] John Lombardo: *Embedded Linux,* New Riders Publishing; 2001; ISBN: 0-7357-0998-X

[67] Craig Hollabaugh, Ph.D.: *Embedded Linux - Hardware, Software, and Interfacing,* Addison Wesley, Inc.; 2002; ISBN: 0-672-32226-9

[68] Bob Smith, John Hardin, Graham Phillips, and Bill Pierce:
LINUX APPLIANCE DESIGN - A Hands-On Guide to Building Linux Appliances, No Starch Press, Inc.; 2007; ISBN: 978-1-59327-140-4

[69] David E. Simon: *An Embedded Software Primer,* Addison Wesley, Inc.; 1999; ISBN: 0-201-61569-X

[70] John Waldron: *Introduction to RISC Assembly Language Programming,* Addison Wesley, Inc.; 1999; ISBN: 0-201-39828-1

[71] Steve Furber: *ARM System Architecture,* Addison Wesley, Inc.; 1996; ISBN: 0-201-40352-8

Online References

Obvious, but to be written due to German-Law:
"It should be recognized, that the given links within this and following sections are solely owned by and are within the exclusive responsibility of their owners. The intention to reference to that sites is neither to take ownership of their work, nor to state commitment to any given statement on their sites. It is much more to honour the work, and thank to the help, the author advanced from by himself. Last but not least, the intention is to support a short cut for the users of UnifiedSessionsManager to sources of required help."

OSs

[72] RedHat(TM) Enterprise Linux: *http://www.redhat.com*

[73] RedHat(TM) Enterprise Linux-Doc:
http://www.redhat.com/docs

[74] CentOS: *http://www.centos.org*

[75] CentOS-Doc: *http://www.centos.org/docs*

[76] Scientific Linux: *http://www.scientificlinux.org*

[77] Debian: *http://www.debian.org*

[78] OpenSuSE: *http://www.opensuse.org*

[79] OpenBSD: *http://www.openbsd.org*
[80] OpenBSD-FAQ: *http://www.openbsd.org/faq/index.html*
[81] OpenBSD-PF: *http://www.openbsd.org/faq/pf/index.html*
[82] FreeBSD: *http://www.freebsd.org*
[83] NetBSD: *http://www.netbsd.org*

[84] uCLinux: *http://www.uclinux.org*
[85] Linux on ARM: *http://www.linux-arm.com*
[86] eCos: *http://ecos.sourceware.org*

Hypervisors/Emulators

kvm

[87] KVM: *http://de.wikipadia.org/wiki/Kernel_based_Virtual_Machine*

QEMU

[88] QEMU(TM): *http://www.qemu.org*
[89] QEMU-CPU support: *http://fabrice.bellard.free.fr/qemu/status.html*
[90] QEMU-User Manual: *http://fabrice.bellard.free.fr/qemu/qemu-doc.html*
[91] QEMU-OS support: *http://www.claunia.com/qemu*
[92] QEMU-Debian: *http://909ers.apl.washington.edu/ dushaw/ARM*
[93] QEMU-Debian on an emulated ARM machine; Aurelien Jarno: *http://www.aurel32.net*
[94] QEMU-Debian kernel and initrd for qemu-arm-versatile; Aurelien Jarno: *http://people.debian.org/ aurel32*
[95] QEMU-Debian ARM Linux on QEMU; Brian Dushaw: *http://909ers.apl.washington.edu/ dushaw/ARM*
[96] QEMU-Ubuntu-Installation/QemuEmulator: *http://help.ubuntu.com/community/Installation/QemuEmulator*
[97] QEMU-Ubuntu-VMwarePlayerAndQemu: *http://wiki.ubuntu.com/VMwarePlayerAndQemu*
[98] QEMU-OpenBSD: *http://141.48.37.144/openbsd/qemu.html - Dag Leine Institut fuer Physikalische Chemie, Martin Luther Universitaet Halle*

[99] QEMU-NetBSD - Running NetBSD on emulated hardware:
http://www.netbsd.org/ports/emulators.html

[100] QEMU-OpenSolaris: *http://www.opensolaris.org/os/project/qemu/host*

[101] QEMU-OpenSolaris Networking:
http://www.opensolaris.org/os/project/qemu/Qemu_Networking

[102] QEMUlator: *http://qemulator.createweb.de*

SkyEye

[103] SkyEye: *http://skyeye.sourceforge.net*

VMware

[104] VMware(TM) Inc.: *http://www.vmware.com*

[105] VMware-OpenBSD - HowTo Install VMWare-Tools:
http://openbsd-wiki.org

[106] VMware-Communities: *http://communities.vmware.com*

[107] VMware-Forum: *http://vmware-forum.de*

[108] VMware-Any-Any-Patch from Petr Vandrovec:
http://kinikovny.cvnt.cz/ftp/pub/vmware

[109] Open Virtual Machine Tools: *http://open-vm-tools.sourceforge.net*

Xen

[110] RED HAT(TM) ENTERPRISE LINUX 5.1 - Virtualization:
http://www.redhat.com/rhel/virtualization

[111] Virtual Machine Manager - VMM:
http://virtual-manager.et.redhat.com

[112] libvirt: *http://www.libvirt.org*

[113] Xen(TM): *http://www.xen.org*

[114] XenWiki - Xen-Networking:
http://wiki.xensource.com/xenwiki/XenNetworking

[115] Xen-CentOS: *Creating and Installing a CentOS5 domU instance*

[116] Xen-Fedora: *http://fedoraproject.org/wiki/FedoraXenQuickstartFC6*

[117] Xen-OpenSolaris: *http://www.opensolaris.org/os/community/xen*

[118] Gerd Hoffmann: *"Install SuSE as Xen guest."*

Security

[119] OpenSSH: *http://www.openssh.org*

[120] SSH Communications Security(TM): *http://www.ssh.com*

[121] SSH Tectia(TM): *SSH Tectia*

[122] OpenSSH-FAQ: *http://www.openbsd.org/openssh/faq.html*

[123] OpenSSL: *http://www.openssl.org*

[124] OpenLDAP: *http://www.openldap.org*

[125] MIT-Kerberos: *http://web.mit.edu/kerberos/www*

[126] Heimdal-Kerberos: *http://www.h5l.org*

[127] sudo - "superuser do"(check google for actual link): *http://www.sudo.ws*

Specials

Dynagen/Dynamips

[128] Dynagen, by Greg Anuzelli:
Dynagen

[129] Dynamips, Cisco(TM) 7200 Simulator:
Dynamips, Cisco(TM) 7200 Simulator

QEMU-Networking with VDE

[130] VDE-Virtual Distributed Ethernet:
http://sourceforge.net/projects/vde

[131] VirtualSquare: *http://virtualsquare.org*

[132] VirtualSquare: *Basic Networking*

PXE

[133] By H. Peter Anvin: *SYSLINUX - PXELINUX - ISOLINUX*

[134] PXE-ROM-Images: *Etherboot*

Routing

[135] "Linux Advanced Routing&Traffic Control", by Bert Hubert:
LARTC-HOWTO

Scratchbox

[136] *Scratchbox*

Serial-Console

[137] By van Emery: *"Linux Serial Console HOWTO"*
[138] By David S. Lawyer / Greg Hawkins: *"Serial-HOWTO"*
[139] By David S. Lawyer: *"Text-Terminal-HOWTO"*

Miscellaneous

[140] IEEE: *http://www.ieee.org*
[141] The GNU Netcat: *Netcat*
[142] Netcat Wikipedia: *Netcat Wikipedia*
[143] By van Emery: *"Linux Gouge"*

UnfiedSessionsManager Versions

[144] The first public version of 2008.02.11, by Arno-Can Uestuensoez. Available with online ASC-II help only by "ctys -H print": *"UnifiedSesionsManager" http://sourceforge.net/projects/ctys*

[145] The second public version of 2008.07.10, by Arno-Can Uestuensoez: *"UnifiedSesionsManager" http://sourceforge.net/projects/ctys*

[146] The second public version with minor updates of 2008.08.6, by Arno-Can Uestuensoez: *"UnifiedSesions-Manager" http://sourceforge.net/projects/ctys*

[147] Minor editorial updates of 2008.08.12, by Arno-Can Uestuensoez: *"UnifiedSesionsManager" http://sourceforge.net/projects/ctys*

[148] Enhanced documentation, 2008.08.16, by Arno-Can Uestuensoez: *"UnifiedSesionsManager" http://sourceforge.net/projects/ctys*

[149] Minor editorial changes, 2009.01.26, by Arno-Can Uestuensoez: *"UnifiedSesionsManager"*

http://sourceforge.net/projects/ctys

Sponsored OpenSource Projects

[150] UnifiedSessionsManager:
http://www.UnifiedSessionsManager.org
http://sourceforge.net/projects/ctys
http://ctys.berlios.de

[151] zamklibant:
http://sourceforge.net/projects/zamklibant

Commercial Support

Commercial support and additional services are available exclusively from:

[152] Arno-Can Uestuensoez:
http://www.uestuensoez.com

[153] Ingenieurbuero Arno-Can Uestuensoez:
http://www.i4p.com