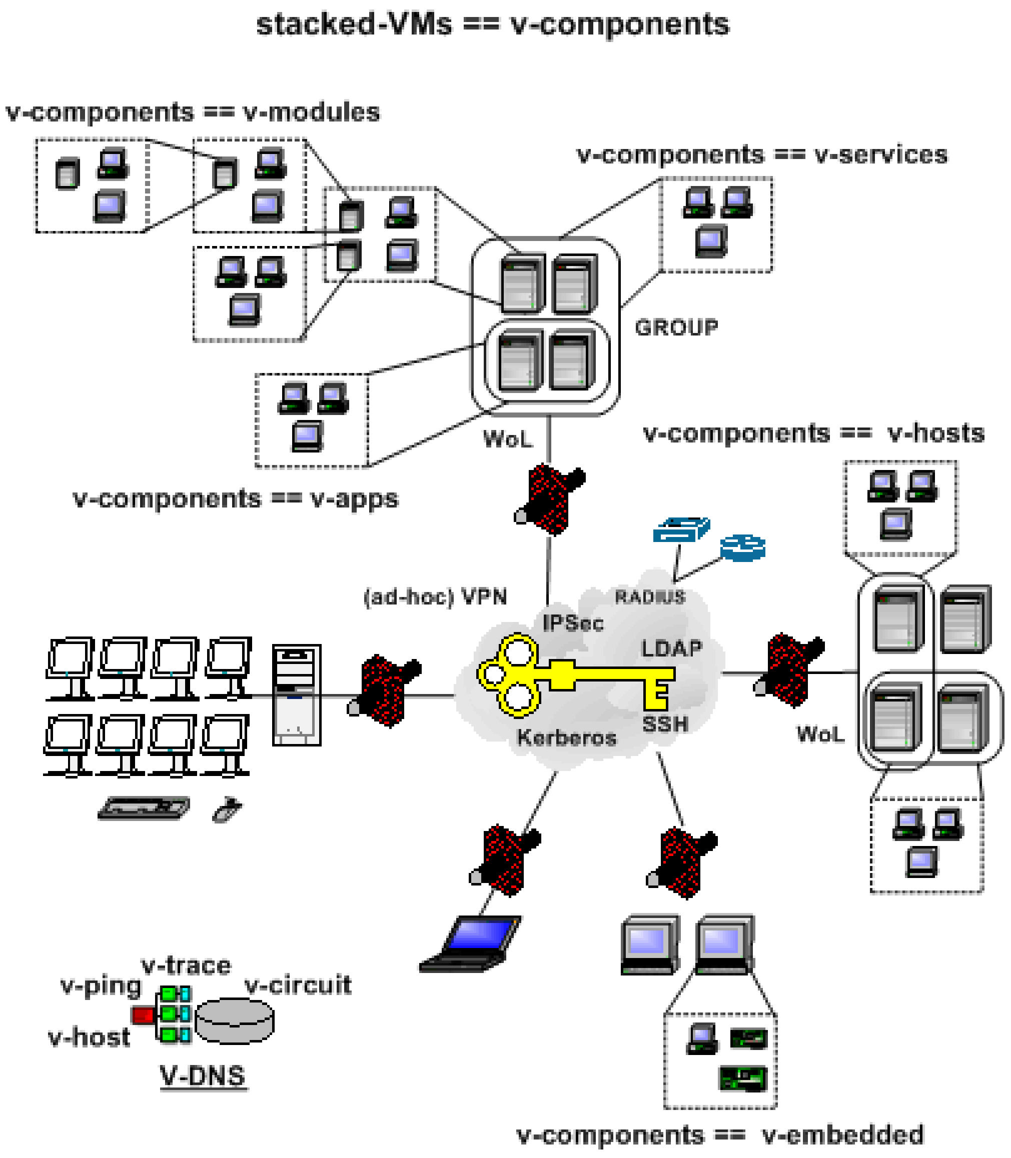


The UnifiedSessionsManager

CTYS-USER-MANUAL
PDF-PRINT-HELP
Version:01.11.020 - 2011.03.02
Release: International

Arno-Can Uestuensoez

Published by
Ingenieurbuero fuer Telekommunikations und Software-Systemloesungen
Arno-Can Uestuensoez

Zentnerstr. 34
D-80798 Muenchen
Germany

Phone:+49.89.27817287

http://www.i4p.com
http://www.UnifiedSessionsManager.org

The following text is required dur to formal reasons, if you are going to use these artifacts for your personal private purposes only you may probably not require to analyse it in detail. The application by scholars, students, and apprentices is specifically desired.

The reason of the introduction of this distinction is the experience of repetitive and ungoing unfair competition with 'criminal elements' contained, having a vast effect on the re-financiation of the further development for this project. Thus this step could be named 'self-defence'.

Software: GPL3

Basic-Documents: GFDL-1.3

Concepts+Interfaces+Documents: CCL - Creative Commons License - Non-Commercial, No-Derivs

Contents

List of Tables

List of Figures

Part I

Common Basics

Chapter 1

Preface

1.1 History

Version	Date	Author	Description
01.03.003.a01[146]	2008.02.11	Arno-Can Uestuensoez	Initial pre-release as embedded printable help
01.07.001.a01[147]	2008.08.03	Arno-Can Uestuensoez	First major update with numerous additions and partial review.
01.07.001.b02[148]	2008.08.11	Arno-Can Uestuensoez	Minor editorial updates. A lot of tests, some fixes.
01.07.001.b03[149]	2008.08.12	Arno-Can Uestuensoez	Minor editorial updates.
01.07.001.b04[150]	2008.08.16	Arno-Can Uestuensoez	Enhancement of documentation and Web-Site.
01.11.001[151]	2010.04.25	Arno-Can Uestuensoez	Major enhancements and updates.
01.11.002	2010.05.24	Arno-Can Uestuensoez	Documentation and web site enhancements.
01.11.003[151]	2010.05.31	Arno-Can Üstünsöz	Patch Default-Port VMware(TM)-Server-2.x, new tool ctys-beamer, add some documentation.
01.11.005[151]	2010.06.27	Arno-Can Üstünsöz	Alpha version of RDP plugin, bugfixes, added some documentation.
01.11.006[151]	2010.07.14	Arno-Can Üstünsöz	Alpha version of VBOX - VirtualBox(TM) plugin, bugfixes, added documentation, preparation of Typo3-Website.
01.11.008[151]	2010.07.30	Arno-Can Üstünsöz	Alpha-Version EnterpriseLinux, bugfixes, added documentation, First Gnome-Menues, ctys-scripts.
01.11.009[151]	2010.08.16	Arno-Can Üstünsöz	Alpha-Version gnome-starter, ctys-config, Fehlerbereinigungen, Ergänzung Dokumentation.
01.11.010[151]	2010.08.20	Arno-Can Üstünsöz	Verify GuetsOSs: ucLinux-QEMU(ARM+Coldfire), QNX-QEMU(x86), QNX-VBOX(x86),bugfixes, added documentation.
01.11.011[151]	2010.11.07	Arno-Can Üstünsöz	Verify New GuetsOSs: Android, MeeGo, RHEL, QNX. Version Updates: CentOS, Debian, OpenSUSE, OpenBSD, Ubuntu. Bugfixes, extension of documentation. menu generation.
01.11.014[151]	2010.11.22	Arno-Can Üstünsöz	Minor editorial.

1.2 Contact

Public maintenance:	acue@sf1_sourceforge.net
Administrative contact:	acue@UnifiedSessionsManager.org acue@UnifiedSessionsManager.eu
Commercial Services: The professional services are offered for end-customers only, so called 'body-leasers' are definetly not welcome.	Engineering Office Arno-Can Uestuensoez - www.i4p.com Ingenieurbuero Arno-Can Uestuensoez - www.i4p.com

1.3 Legal

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

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

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

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

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

QEMU is a trademark of Fabrice Bellard.

All mentioned RealVNC products and their registered names are Trademarks of the Company RealVNC Ltd.

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

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

Xen is a trademark of XenSource Inc.

If some is forgotten, it will be added immediately.

1.4 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 intermixed operations and management of local and remote sessions on physical and virtual machines.

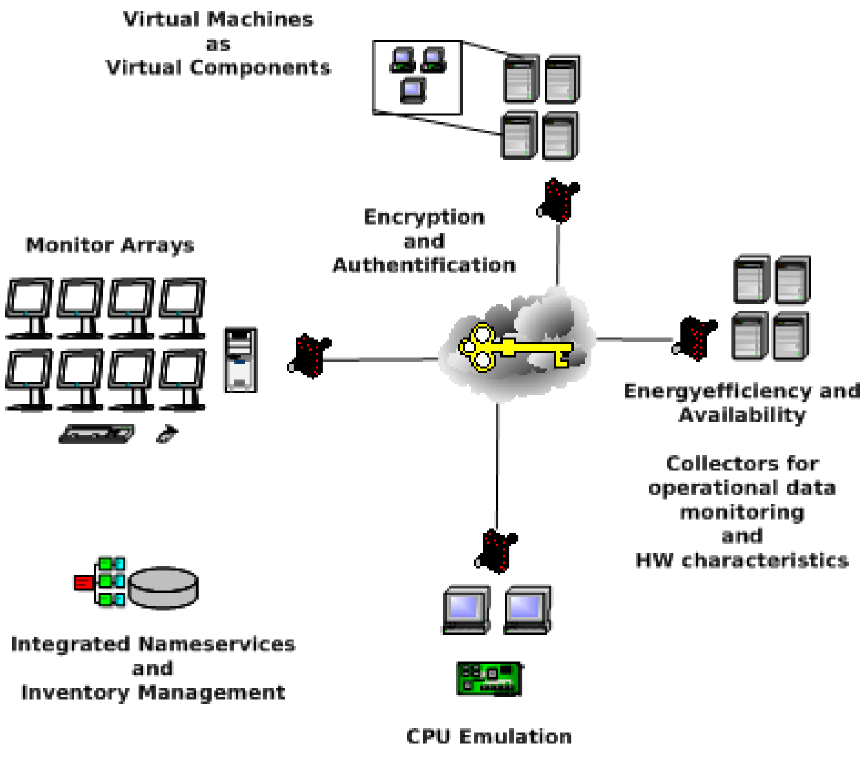


Figure 2.1: The UnifiedSessionsManager

The primary target was to combine facilities for the management of physical and virtual machines including the modelled sessions objects - alltogether combined with networking and security features - into a seamless interface.

- Management of Nested Multi-Level Stacks of Virtual Machines as Virtual Components
- Management of User-Interfaces on 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. The main building block for this is the handling of the desktop presentation of the managed entities. This particularly comprises the handling of session windows on an X11 based desktop with logically combined screens by the so called "Xinerama" mode. But also some addressing facilities for disconnection and re-establishment of sessions to headless-running server entities.

- Support of seamless 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.

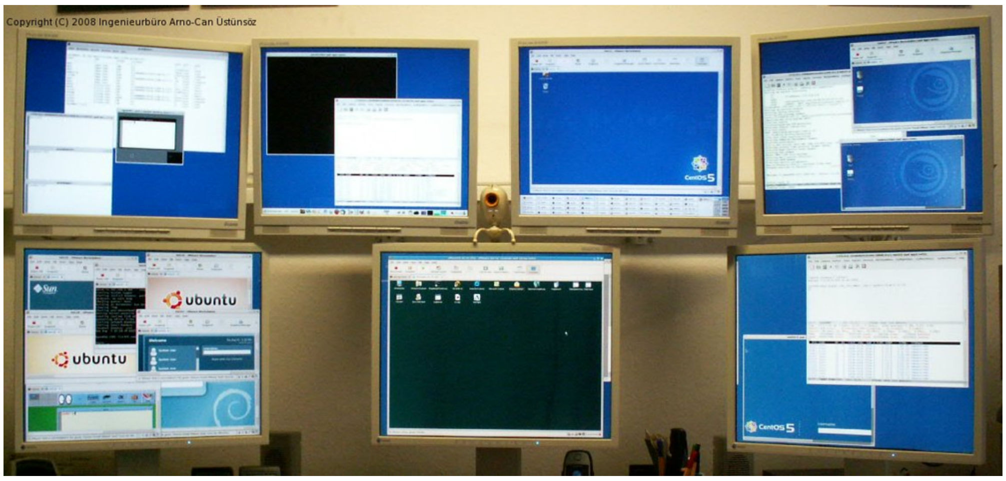


Figure 2.2: Physical Multi-Monitor Design

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.

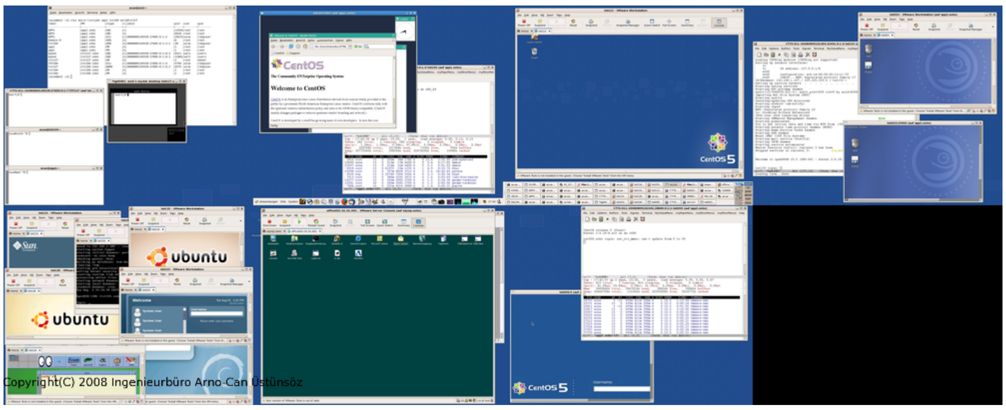


Figure 2.3: Logical Xinerama-Mode

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.

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.

Current provided standard components are CLI , X11 , VNC , KVM (accelerator for QEMU), QEMU , VMW (VMware-Workstation/Server/Player), XEN , and PM (Linux, Solaris, OpenSolaris, OpenBSD, and FreeBSD). Any OS is supported, when control by hypervisor only is sufficient.

VirtualBox and OpenVZ are going to be intergrated next, as well as the specific upgrades for Server editions ov XEN and VMW.

Chapter 3

Feature Specification

3.1 Feature Introduction

The "UnifiedSessionsManager" comprises a number of first-time implemented features assembling to a solution for configuration and operation of environments with bulks of virtual and physical processing nodes.

Some to be mentioned are:

1. Management of User-Interfaces on Distributed Monitor-Arrays
2. Management of nested multi-level virtualizations
3. Support of Integrated CPU-Emulation for multiple architectures
4. Definition on an extended Address Schema
5. Support of Integrated Nameservices
6. Built-In support fo Encryption and Authentication
7. Introduction of GROUPS concept

3.2 Feature-Sum-Up

The following tables present an overview of the supproted components for current release. The listed PC, Workstation and Server based platforms with listed Hypervisors are supported and tested when marked with "OK". Additional platforms are going to be added for next versions("*").

The utility "ctys-genmconf" supports the detection and generation of relevant control data, the utility "ctys-plugins" verifies actual available operational states and resultingfeatures.

The main development and production platform for the UnifiedSessionsManager is CentOS.

The following pages show the current operational and test states of the various combinations of hypervisors, HostOS, and GuestOS. The actual operational states are visualized by specific colors as shown in next table.

Color	State
🟪	The versions actually targeted to be supported with maximum available feature set.
🟩	OK: Tested and operational in current release.
🟦	NEXT: Sceduled for the next release. Probably already partly tested.
*	PLANNED: Intended for a later release.
-	OPEN: Technically possible, but for some reasons not yet planned to be implemented.

Table 3.1: Color coding of implementation and test states.

3.2.1 Supported Hypervisors

Supported Hypervisors on platforms as shown in the following tables.

Plugin	Supported Hypervisor	Versions		
		Previous	Current	InProcess
🟪 KVM	KVM		🟩 2.6.18/kvm-83 2.6.18-6/kvm-62 2.6.26-1/kvm-72	
OVZ	OpenVZ			
🟪 QEMU	Qemu	🟩 0.9.0	🟩 0.9.1,11.0.0,0.12.2	🟦 0.12.3
🟪 VBOX	VirtualBox(TM)		🟩 3.1.2	🟦 3.2.8
VMW	VMware-Player(TM)	🟩 1.0.4	🟩 1.0.5,2.5.3,3.0.1	
🟪 VMW	VMware-Server (TM)	🟩 1.0.4,1.0.6,1.0.9	🟩 1.0.10,2.0.2	
🟪 VMW	VMware-Workstation(TM)	🟩 6.0.2,6.0.4,6.5.1	🟩 6.5.3,7.0.1	
VMW	VMware-ESXi-Server(TM)			🟦 4.x.x
VMW	VMware-ESX-Server(TM)			🟦 4.1.0
🟪 XEN	Xen(TM)		🟩 3.0.3,3.1.0	🟦 3.3.0,3.3.1,3.4.2
XEN	Citrix-XenServer(TM)			🟦 5.5.0, 5.6.0

Table 3.2: Supported Hypervisors

3.2.2 Tested GuestOS support

The following table lists the already tested OS-Distribution vs. Containing Plugins. The containing plugins comprise the plugin itself as well as the required software and hypervisors.

Distribution	PMs		VMs						
	0	1+n	KVM	OVZ	VBOX	VMW	XEN	QEMU	
	x86	x86	x86	x86	x86	x86	x86	x86	ARM
					BSD				
FreeBSD-7	OK	*	OK	-	*	OK	-	OK	-
FreeBSD-8	OK	*	OK	-	*	OK	-	OK	-
NetBSD-5.2	*	*	*	-	OK	*	*	*	*
OpenBSD-4[1]	OK	OK	OK	-	OK	OK	-	OK	-
					Linux				
CentOS-5	OK	OK	OK	*	OK	OK	OK	OK	-
Debian-4-etch	OK	OK	-	-	-	OK	-	OK	-
Debian-5-lenny	OK	OK	OK	*	OK	OK	OK	OK	*
Enterprise Linux Server 5/ Unbreakable Linux	*	*	OK	*	OK	*	-	OK	-
Fedora-8	OK	OK	OK	-	-	OK	OK	OK	-
Fedora-10	OK	OK	OK	-	-	-	-	OK	-
Fedora-12	-	-	-	-	OK	-	-	-	-
Fedora-13	*	*	OK	-	OK	*	-	OK	-
Knoppix6.2	*	*	OK	-	*	*	*	OK	-
Knoppix-6.2.1 ADRIANE	OK	OK	OK	-	OK	*	*	OK	-
Mandriva-2010	OK	OK	OK	-	OK	OK	*	OK	-
Scientific Linux	OK	OK	OK	-	OK	OK	*	OK	-
openSUSE-10.3	OK	OK	-	-	-	OK	-	OK	-
openSUSE-11.1							OK		
openSUSE-11.2	OK	OK	OK	-	OK	OK	*	OK	-
openSUSE-11.3	OK	OK	OK	-	OK	*	*	OK	-
RedHat-Enterprise Linux 5	OK	OK	OK	-	OK	X	X	OK	-
RedHat-Enterprise Linux 6beta	OK	OK	OK	-	OK	X	X	OK	-
Slackware-13.1	*	*	*	-	*	*	*	*	*
SuSE-9.3	-	OK	-	-	-	OK	-	-	-
SuSE-10.2	-	OK	-	-	-	-	OK	-	-
Ubuntu-6.06.1-dapper	-	OK	-	-	-	OK	-	-	-
Ubuntu-7.10-gutsy	-	OK	-	-	-	OK	-	-	-
Ubuntu-8.04-hardy	OK	OK	OK	-	-	OK	OK	OK	-
Ubuntu-9.10	OK	OK	OK	*	OK	OK	*	OK	-
Ubuntu-10.10	OK	OK	OK	*	OK	OK	X	OK	-

Table 3.3: Getestete GuestOS

Distribution	PMs		VMs						
	0	1+n	KVM	OVZ	VBOX	VMW	XEN	QEMU	
	x86	x86	x86	x86	x86	x86	x86	x86	ARM
Solaris(TM)									
Solaris-10[2]	OK	OK	OK	-	OK	OK	-	OK	-
OpenSolaris-2009.6[4]	-	OK	OK	-	OK	OK	*	OK	-
ILLUMOS[5]	-	*	*	-	*	*	*	*	-
Nexenta[6]	-	*	*	-	*	*	*	*	-
OpenIndiana[7]	-	*	*	-	*	*	*	*	-
DOS									
FreeDOS[7]	-	-	*	-	*	*	*	OK	-
Balder[7]	-	-	*	-	*	*	*	OK	-
MS-Dos-5.x[7]	-	-	*	-	*	*	*	*	-
MS-Dos-6.x[7]	-	-	*	-	*	*	*	*	-
Windows									
MS-Windows-NT[7]	-	-	*	-	*	OK	*	*	-
MS-Windows-2000[7]	-	-	*	-	*	OK	*	*	-
MS-Windows-XP[7]	-	*	*	-	OK	OK	*	*	-
MS-Windows-2003[7]	-	*	*	-	*	OK	*	*	-
MS-Windows-7[7]	-	*	*	-	OK	*	*	*	-
MS-Windows-2008[7]	-	*	*	-	OK	*	*	*	-
Smartphone									
Android-2.2	*	*	OK	-	OK	-	-	OK	*
MeeGo-1.0	*	*	(X)	-	OK	*	-	(X)	*
Embedded									
FreeRTOS	*	*	-	-	-	-	-	-	*
QNX	*	*	*	-	OK	-	-	(OK)	*
uCLinux	*	*	-	-	-	-	-	*	(OK)

Table 3.4: Tested GuestOS

[5]No WoL for now.

[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.

[7]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.

3.2.3 Supported Native Plugins

The next table shows the passed tests of supported native plugins vs. OS-Distribution. The plugins including required hypervisors are to be executed on the listed OSs. Other OSs and versions might work as well.

Distribution	PMs	VMs						HOSTs			
	PM	KVM	OVZ	QEMU	VBOX	VMW	XEN	CLI	RDP	VNC	X11
BSD											
FreeBSD-7	OK		-	*	-	-	*	OK	*	OK	OK
FreeBSD-8	OK		-	*	-	-	*	OK	*	OK	OK
NetBSD-5.2	*		-	*	-	-	*	*	*	*	*
OpenBSD-4	OK	-	-	X	-	-	-	OK	*	OK	OK
Linux											
CentOS-5	OK	OK	*	OK	OK	OK	OK	OK	OK	OK	OK
Debian-4-etch	OK	-	-	OK	-	OK	-	OK		OK	OK
Debian-5-lenny	OK	OK	*	OK	OK	OK	OK	OK	OK	OK	OK
Enterprise Linux Server 5 / Unbreakable Linux	OK	OK	*	OK	X	-	OK	OK	(OK)	OK	OK
Fedora-8	OK	-	-	OK	-	-	OK	OK		OK	OK
Fedora-12	OK	*	-	*	*	*	*	OK		OK	OK
Fedora-13	*	*	-	*	*	*	*	OK	(OK)	OK	OK
Gentoo	*	-	-	-	-	-	-	*	*	*	*
Knoppix	OK	*	-	*	*	*	*	OK	(OK)	OK	(OK)
Mandriva-2010	OK	-	-	-	-	-	-	OK		OK	OK
openSUSE-10.3	OK	-	-	OK	-	OK	-	OK		OK	OK
openSUSE-11.1	OK	-	-	-	-	-	OK	OK		OK	OK
openSUSE-11.2	OK	OK	-	OK	*	*	*	OK		OK	OK
openSUSE-11.3	OK	OK	*	OK	OK	*	OK	OK	OK	OK	OK
RedHat-Enterprise Linux 5.5	OK	OK	*	OK	*	OK	OK	OK	OK	OK	OK
RedHat-Enterprise Linux 6.0 beta	OK	X	*	X	*	*	*	OK	OK	OK	OK
Scientific Linux SL 5.4.1	OK	OK	-	OK	*	OK	OK	OK	*	OK	OK
Sackware-13.1	*	*	-	*	*	*	*	*	*	*	*
SuSE-9.3	OK	-	-	-	-	OK	-	OK		OK	OK
SuSE-10.2	OK	-	-	-	-	-	-	OK		OK	OK
Ubuntu-6.06.1	-	-	-	-	-	-	-	OK		OK	OK
Ubuntu-7.10	-	-	-	-	-	-	-	OK		OK	OK
Ubuntu-8.04	OK	OK	-	(OK)[8]	-	-	-	OK		OK	OK
Ubuntu-9.10	OK	*	-	*	*	*	*	OK	*	OK	OK
Ubuntu-10.10	OK	OK	*	OK	X	X	X	OK	OK	OK	OK

Table 3.5: Native Plugins vs. OS-Distribution

Distribution	PMs	VMs						HOSTs			
	PM	KVM	OVZ	QEMU	VBOX	VMW	XEN	CLI	RDP	VNC	X11
Hypervisor-Distributions											
ESXi	*	-	-	-	-	-	-	*		*	-
ESX-4.1.0	X	-	-	-	-	X	-	OK	X	OK	OK
XenServer-5.5.0[9]	X	-	-	-	-	-	X	OK	X	OK	OK
Solaris(TM)											
Solaris-10	(OK)	-	-	-	-	-	-	(OK)		(OK)	(OK)
OpenSolaris 2009.6	OK	-	-	X	X	-	*	OK	X	OK	OK
ILLUMOS	*	-	-	*	*	-	*	*	*	*	*
Nexenta	*	-	-	*	*	-	*	*	*	*	*
OpenIndiana	*	-	-	*	*	-	*	*	*	*	*
MS-Windows(TM)											
MS-W2K	*	-	-	-	-	-	-	*		*	-
MS-WXP	*	-	-	-	-	-	-	*		*	-
MS-W2K3	*	-	-	-	-	-	-	*		*	-
MS-W2K8	*	-	-	-	-	-	-	*		*	-
Smartphone											
Android-2.2	*	-	-	-	-	-	-	*	*	*	*
MeeGo-1.0	OK	*	-	*	-	-	-	OK	*	*	*
Embedded											
FreeRTOS	*	-	-	-	-	-	-	*	*	*	*
QNX	*	-	-	-	-	-	-	*	*	*	*
RTEMS-Dev	-	-	-	*	*	-	-	*	*	*	*
uCLinux	*	-	-	-	-	-	-	*	*	*	*

Table 3.6: Native Plugins vs. OS-Distribution

[9] Compilation of 'qemu-system-x86_64' with support for '-name' option required.

Unterstützte Produkte und Versionen für die jeweiligen Plugins. Diese varirieren z.T. für die verschiedenen Plattformen.

Plugin / Toolset	Unterstütztes Produkt	Versionen		
		Vorversion	Aktuell	InBearbeitung
CLI	bash		3.2.39.1, >3.x	
RDP	rdesktop		1.6	
	krdc		ffs.	
	tsclient		ffs.	
VNC	RealVNC		3.x	
			4.1.1, 4.1.2	
	TigerVNC		x.x	
	TightVNC		x.x	
	krdc		ffs.	
	tsclient		ffs.	
X11	gnome-terminal		x.x	
	xterm		x.x	
	emacs		21.x, 22.x	

Table 3.7: Unterstützte HOSTs Plugins

Plugin / Toolset	Unterstütztes Produkt	Versionen		
		Vorversion	Aktuell	InBearbeitung
Desktop	Gnome		x.x	
	KDE		x.x	
	fvwm		x.x	
	xfce		x.x	
Shells	bash		3.2.39.1, >3.x	

Table 3.8: Unterstützte HOSTs-Plugin Sub-Komponenten

Plugin / Toolset	Unterstütztes Produkt	Versionen		
		Vorversion	Aktuell	InBearbeitung
QEMU	Qemu	0.9.0	0.9.1,0.11.0,0.12.2	0.12.3
	KQEMU			
	KVM			
VBOX	VirtualBox(TM)		3.1.2	3.2.8, 3.2.10
VMW	VMware-Player(TM)	1.0.4	1.0.5,2.5.3,3.0.1	
	VMware-Server (TM)	1.0.4,1.0.6,1.0.9	1.0.10,2.0.2	
	VMware-Workstation(TM)	6.0.2,6.0.4,6.5.1	6.5.3,7.0.1	
XEN	Xen(TM)		3.0.3,3.1.0	3.3.0,3.3.1,3.4.2,4.0.0

Table 3.9: Unterstützte Server basierte VMs plugins

Plugin / Toolset	Unterstütztes Produkt	Versionen		
		Vorversion	Aktuell	InBearbeitung
VMW	VMware-ESX-Server(TM)		4.1.0	
	VMware-ESXi-Server(TM)			4.0.0
XEN	Citrix-XenServer(TM)		5.5.0	5.6.0

Table 3.10: Unterstützte Host basierte VMs plugins

3.2.4 Tested Client OSs

The following table lists the already tested client OSs.

Distribution	ctys			GUI						
				X11		WM				
	GROUP	DF	CF	X11	Xinerama	Gnome	KDE	fvwm	xfce	ffs.
BSD										
FreeBSD-7										
FreeBSD-8	X	X	X	X	*	X	X	X	-	
NetBSD-5.2	X	X	X	X	*	X	X	X	-	
OpenBSD-4	X	X	X	X	*	X	X	X	-	
Linux										
CentOS-5	OK	OK	OK	OK	OK	OK	X	X	X	
Debian-5-lenny	OK	OK	OK	OK	OK	OK	X	X	X	
Enterprise-Linux Server	*	*	*	*	*	*	*	*	*	
Fedora-8										
Fedora-10	X	X	X	X	*	X	X	X	X	
Fedora-12	*	OK	*	OK	*	OK	*	-	*	
Fedora-13	*	*	OK	OK	*	OK	*	-	*	
Knoppix	X	X	OK	OK	*	OK	X	X	X	
Mandriva-2010	*	OK	*	OK	*	OK	*	*	*	
Scientific Linux	OK	OK	OK	OK	*	OK	OK	-	-	
openSUSE-11.2	OK	OK	OK	OK	*	OK	OK	OK	OK	
openSUSE-11.3	*	*	*	*	*	*	*	*	*	
RedHat-Linux Server 5.5	*	OK	*	*	*	OK	*	*	*	
RedHat-Linux Server 6.0 beta	*	*	*	*	*	*	*	*	*	
Ubuntu-6.06.1-dapper										
Ubuntu-7.10-gutsy										
Ubuntu-8.04-hardy	OK	OK	(OK)	OK		OK	OK	OK	OK	
Ubuntu-9.10	X	X	X	X	X	X	X	X	X	
Ubuntu-10.10	X	X	X	X	X	X	X	X	X	
Hypervisor-Distributions										
ESXi										
ESX	X	X	X	X	*	X	X	X	X	
XenServer-5.5.0	X	OK	X	OK	*	OK	X	X	OK	

Table 3.11: Getestete ClientOS

Distribution	ctys			GUI						
	GROUP	DF	CF	X11		WM				
				X11	Xinerama	Gnome	KDE	fvwm	xfce	ffs.
Solaris(TM)										
Solaris-10	*	*	*	*	*	*	*	*	*	
OpenSolaris-2009.6	X	X	X	X	*	X	X	X	X	
ILLUMOS	*	*	*	*	*	*	*	*	*	
Nexenta	*	*	*	*	*	*	*	*	*	
OpenIndiana	*	*	*	*	*	*	*	*	*	
Windows										
MS-Windows-NT										
MS-Windows-2000	*	*	*	*	*	*	*	*	*	
MS-Windows-XP	*	*	*	*	*	*	*	*	*	
MS-Windows-200x	*	*	*	*	*	*	*	*	*	
Smartphones										
Android	*	*	*							*
MeeGo	*	*	*							*
Embedded										
QNX	*	*	*	-	-	-	-	-	-	*
uCLinux	*	*	*	-	-	-	-	-	-	*
FreeRTOS	*	*	*	-	-	-	-	-	-	*

Table 3.12: Getestete ClientOS

[8] Kein WoL.

[9] Einige Einschränkungen bei LIST.

[10] Unter ausschließlicher Kontrolle des hypervisors. Getested mit diversen Versionen, z.B. Windows-NT-Server, Windows-2000, und Windows-XP.

Chapter 4

Claimed Inventions

Related to IP/SW-Patents it has to be mentioned now, that this software and the implemented concepts were first released on January 2008. The software as a result of my self-sponsored work, is owned solely by myself and donated to the public based on GPL3.

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 future processing power enhancement this heavily seem to become 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 of eCos or 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.

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 114).

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 seamless 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 [133, 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.

4.3 Third set - 2010.05.12

Dynamic Desktop Assembly by generic Addressing

The dynamic address schema of the UnifiedSessionsManager - particularly the LABEL feature - supports for persistent storage of runtime DISPLAY identifiers, thus it is possible in a easy to use and generic manner to set up static pre-configurations as well as add dynamically GUI elements to a running desktop.

This is particularly applied by the two approaches:

1. suboption-VNC-CREATE: VNCDESKIDLIST
   Setting up preconfigured elements of a desktop created by VNC could be used to dynamically assemble required elements - and remove them - as required. This has particularly advances for setup of complex environments in development and test environments, where complete test-cases could be setup by on-demand assembly of their runtime components.

For VNC this is utilised by a two level approach:

(a) Setup desktop-elements within 'HOME/.vnc/xstartup'.

(b) Use call interface of VNC plugin with suboption VNCDESKIDLIST for CREATE.

2. option: '-D <DISPLAY|LABEL>'
Setting a local-only new DISPLAY for output for each call. Particularly applied to the VNC plugin by any GUI-Producing plugin.

The particular advance is the persistency support for arbitrary dynamically allocated desktops by support of the LABEL feature. The LABEL is evaluated and mapped to a numerical DISPLAY value at runtime, when called. Thus this provides a simple facility for preconfigured desktops by usage LABELs for addressing only, this could be applied to GROUP and MACRO feature.

Full-Custom-Installer for VM-Setup - Client-Aware

The setup-tool 'ctys-createConfVM' provides a complete setup resulting in from-the-box ready-to-use VMs, particularly including all relevant client information for the off-line inventory management of the complete VM.

This includes the assignment of required MAC and IP-Addresses fully automated from the integrated database by usage of 'ctys-vhost' and/or 'ctys-macmap'. These tools could be used by fully-automated extracting the required data by usage of 'ctys-extractMAClst' or 'ctys-extractARPlst'.

The full-custom approach is currently applicable for the QEMU/KVM and XEN plugins.

Any additional

- designed and implemented within the current version.

4.4 Third set - 2010.05.31

Integrated addressing of multihop-tunnels

The addressing scheme is extented to the multi-hop routing of relay based connections for automatic path calculation and basically routing-protocol independent interconnection. This schema particularly integrates into the ctys-addressing, where e.g. multi-target distribution of one execution call by usage of GROUPs is supported.

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 management of the sessions client windows on X11 based desktops is supported broadly, by the extension of the X11-geometry option. This comprises the geometry parameters as well as the intuitive addressing of sessions by their visible GUI-Windows-Titles.
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
```

The 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.

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 CONNECTIONFORWARDING".

In addition some 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.

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 additional information refer to "-b" option.

5.1 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 version. 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.

5.1.1 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.

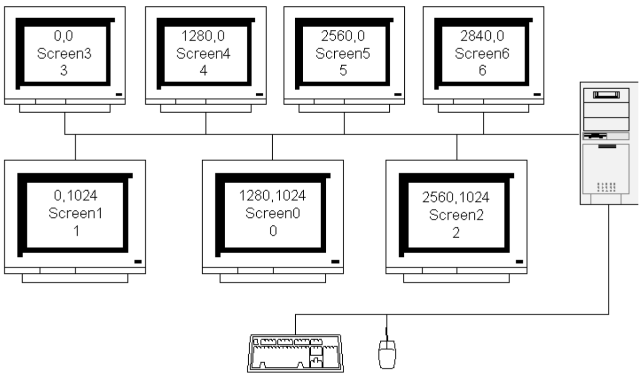


Figure 5.1: Physical Multi-Monitor Layout-1

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.

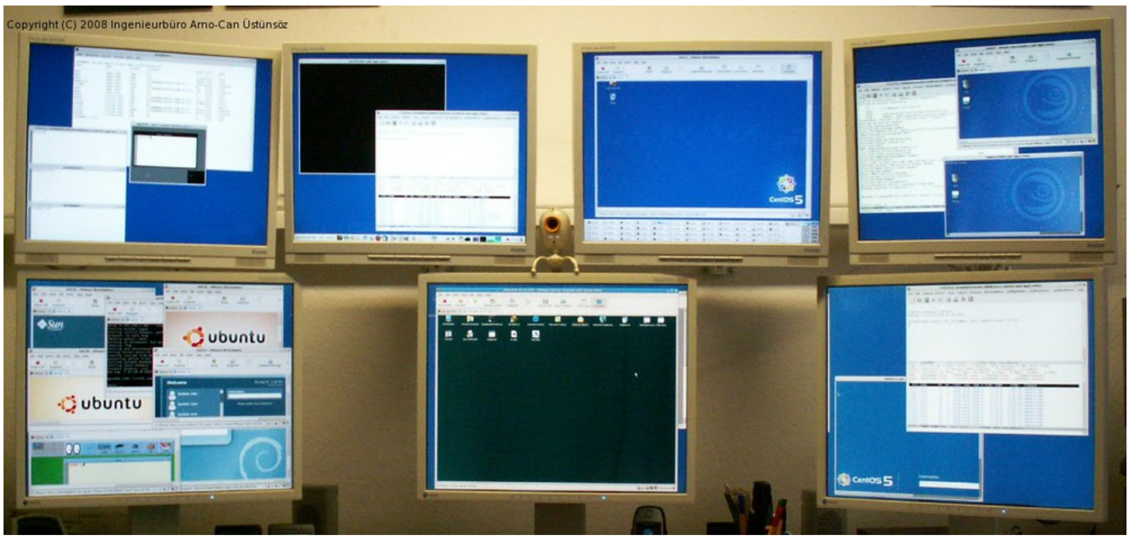


Figure 5.2: Physical Multi-Monitor Design

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.

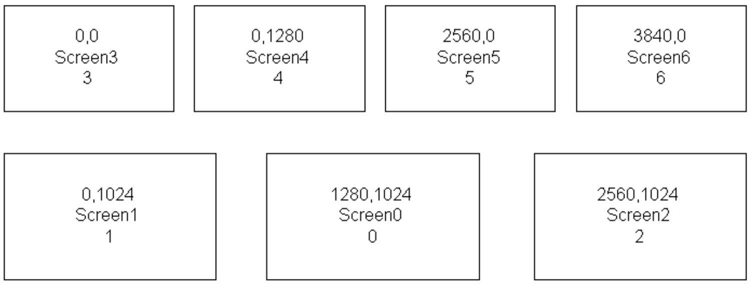


Figure 5.3: Logical Multi-Screen Layout-1

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.

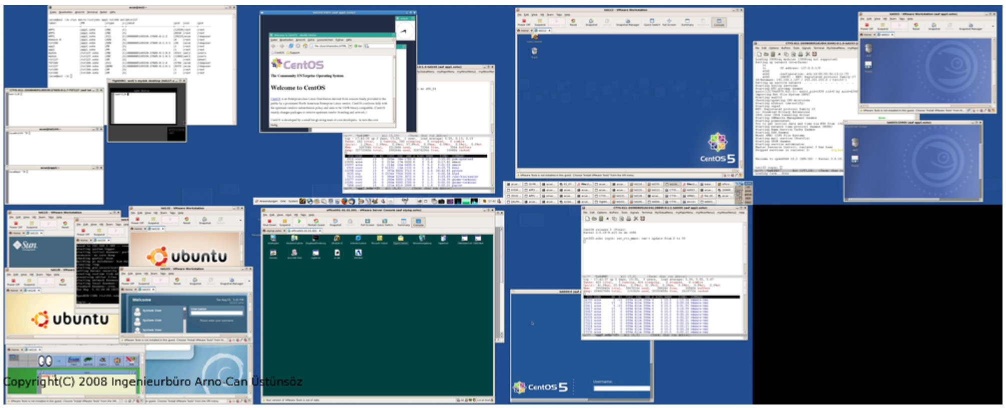


Figure 5.4: Logical Xinerama Layout

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 images 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.

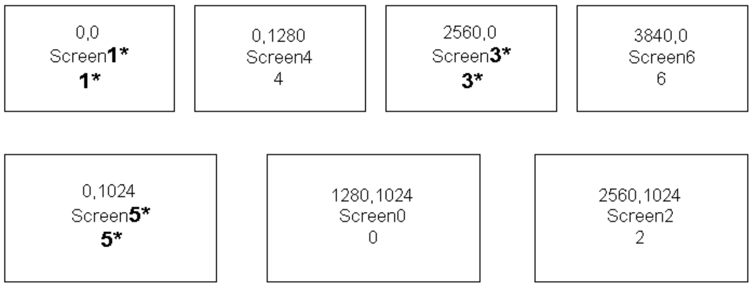


Figure 5.5: Logical Multi-Screen X11-Remapping

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.

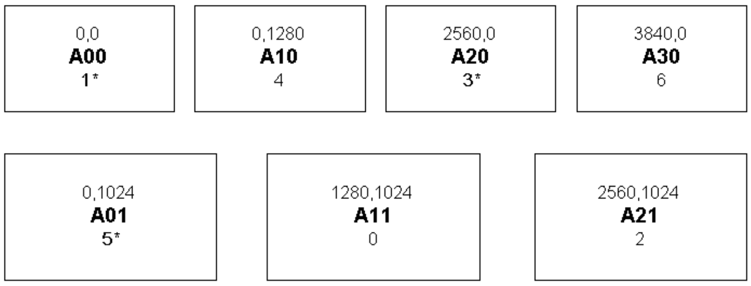


Figure 5.6: Logical Multi-Screen X11-Array-Style

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

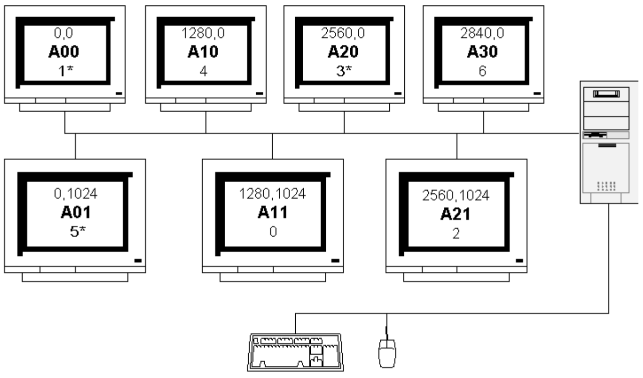


Figure 5.7: Physical Multi-Monitor Array-Style Addressing

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.

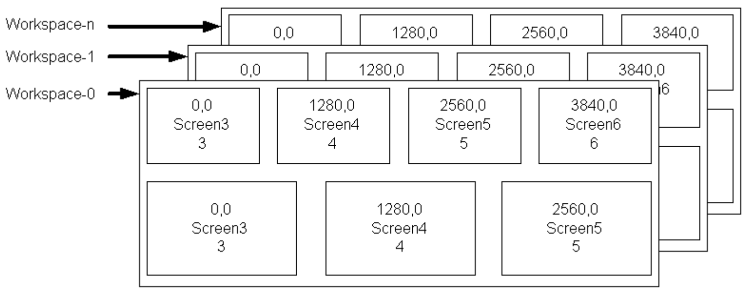


Figure 5.8: Mapping schema for multiple Desktops/Workspaces

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.

5.1.2 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.

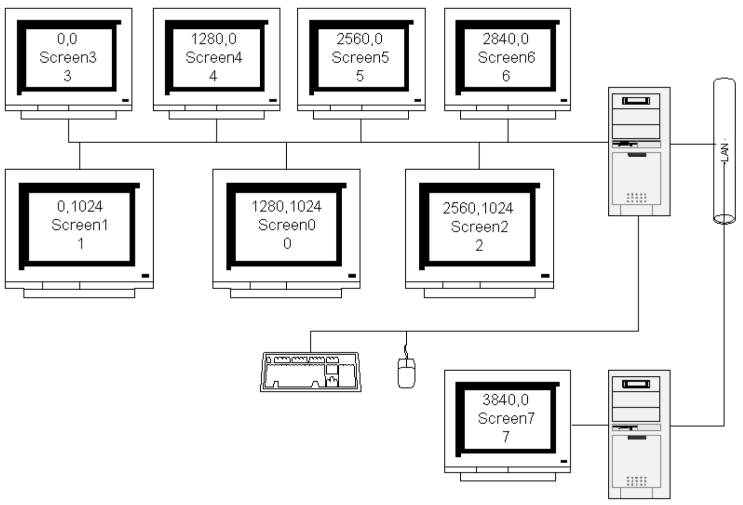


Figure 5.9: Physical Multi-Monitor Layout-2

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.

5.2 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.

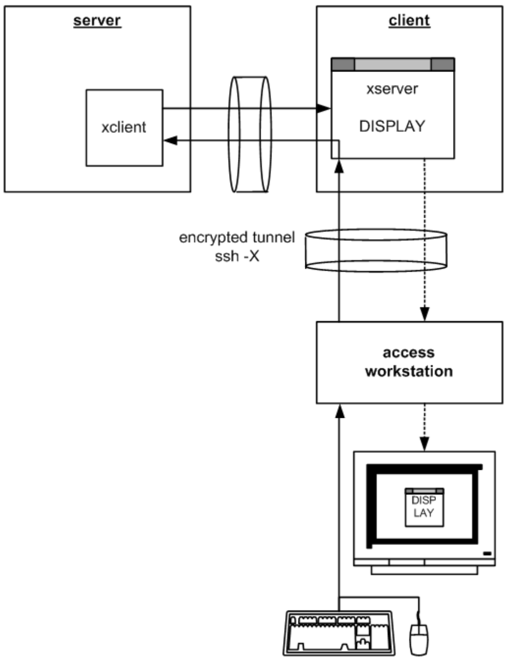


Figure 5.10: Basic handling of client sessions windows

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 Server-Layout, 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.

5.3 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 for each target.

```
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 -W 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.

5.4 Encryption and Tunneling with SSH

Due to security reasons OpenSSH is the only supported connection type. This provides the seamless integration into KerberosV as well as the integration with X11-based user interfaces by providing connections via encrypted tunnels for remote displays.

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

5.4.1 DISPLAYFORWARDING

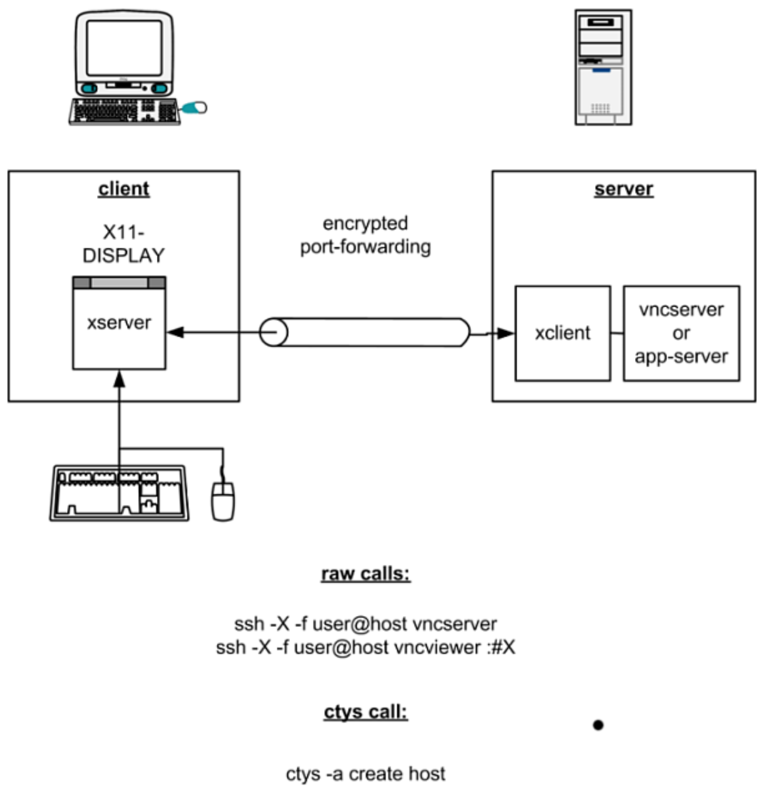


Figure 5.11: DISPLAYFORWARDING

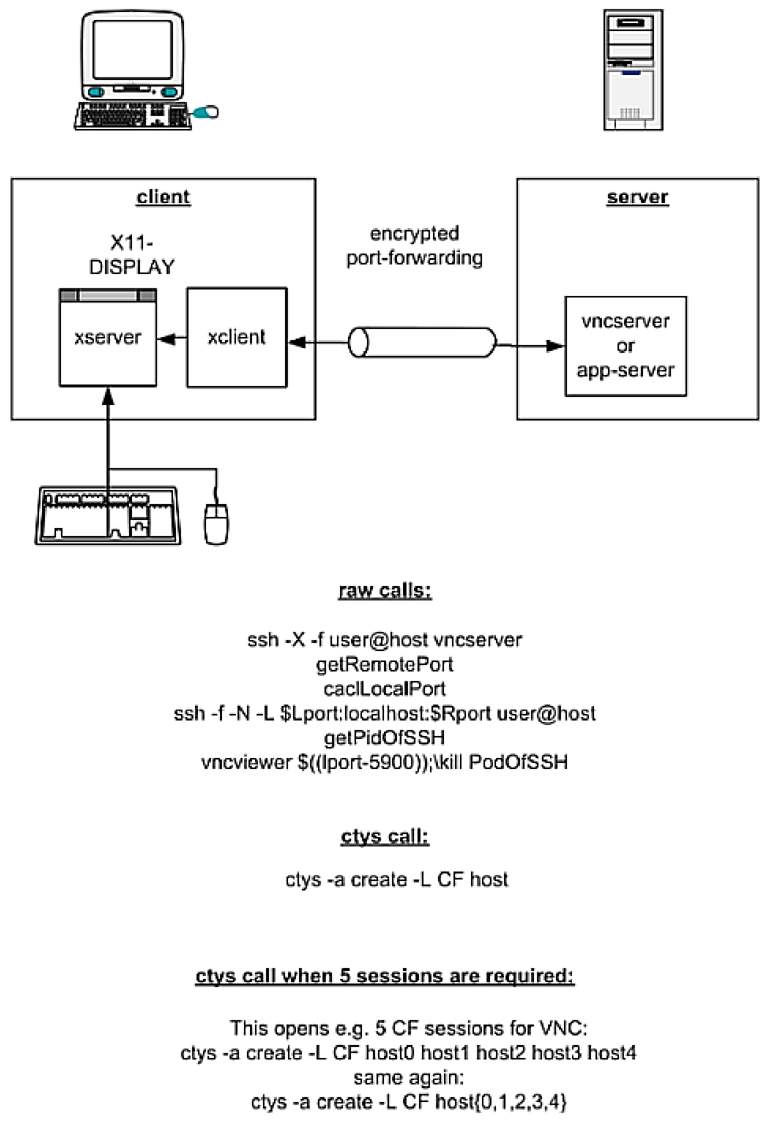
DISPLAYFORWARDING

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.

5.4.2 CONNECTIONFORWARDING



raw calls:

```
ssh -X -f user@host vncserver
getRemotePort
caclLocalPort
ssh -f -N -L $Lport:localhost:$Rport user@host
getPidOfSSH
vncviewer $((lport-5900));\kill PodOfSSH
```

ctys call:

```
ctys -a create -L CF host
```

ctys call when 5 sessions are required:

This opens e.g. 5 CF sessions for VNC:

```
ctys -a create -L CF host0 host1 host2 host3 host4
```

same again:

```
ctys -a create -L CF host{0,1,2,3,4}
```

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.

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

5.4.3 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 **??** '**??**' on page **??** .

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.

6.5 Custom Desktops - Pre- and Post-Configuration

Desktops as the main forcus and possibly most important building block of a User Interface are supported as a portable object class by it's own. In current version this is based on VNC only. The desktop object is defined within the ctys as a collection of coupled entities, founding an aspect oriented workspace. This workspace itself could be combined with additional workspaces into a superior aspect.

The main container object class for GUI elements is currently the VNC plugin. The VNC desktop itself could be assembled dynamically with additional GUI elements by usage of the related DISPLAY variable to the targeted entity.

The following two forms of assembly are supported:

1. Semi-Dynamic Pre-Configuration
   This comprises a custom configuration of the installed file **xstartup** with a 'case' element in shell syntax, and additionally the **CREATE** suboption **VNCDESKIDLIST**. The latter defines a list of user-defined 'case' elements, which are executed during startup by the 'vncserver' with a shell-subcall. The installed xstartup contains a list with various operational examples.

   This style of configuration provides facilities for pre-definition of a set of static GUI-modules which could be combined by the applied **VNCDESKIDLIST** suboption.
2. Dynamic Post-Configuration
   The dynamic post-configuration utilizes the **-D** option, which provides the setting of the DISPLAY variable. The provided values could be either in native X11 syntax for local addressing - without the leading colon, or a ctys LABEL could be applied. The advance of the apllication of a LABEL ist the runtime independency, because the dynamic assertion of DISPLAY IDs my vary from call to call. The usage of LABELs provides the definition of independent MACROs.

Refer to the HOWTO-Manual for concrete examples.

Chapter 7

HOSTs - Native Access

Host sessions are general purpose sessions connecting a user with an arbitrary active instance of an OS. A HOSTs session is actually a login with some additional processes for remote access. The available standard HOSTs sessions are CLI, X11, RDP, and VNC. All of these are supported to be accessed via SSH encryption only. The callee is executed by ssh encapsulation.

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 remote command execution is supported by specific suboptions.

7.2 Start a GUI application - X11

The X11 plugin supports the start of GUI based terminals or in general GUI based applications. Therefore any command could be executed on the remote 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.

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 required remote ports could be statically customized or dynamically allocated.

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.

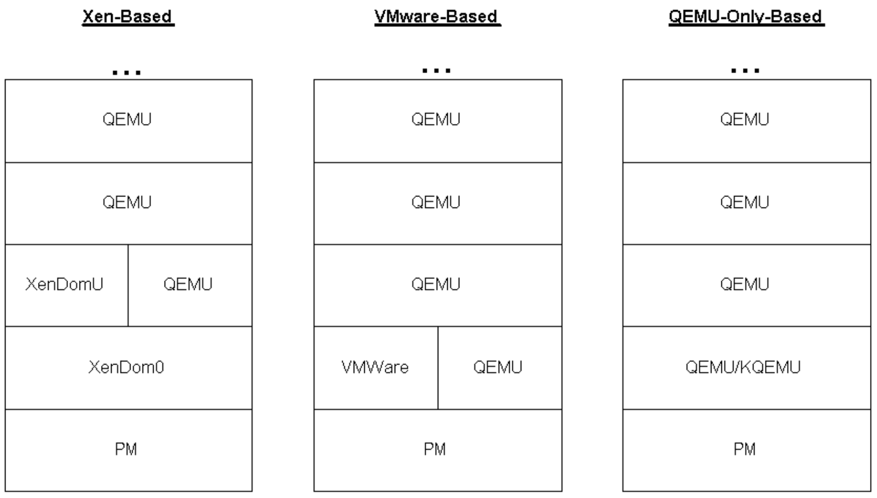


Figure 8.1: Supported Stackmodels

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.
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.

PM/VM	Description
CLI	Command Line Interface.
X11	X11 caller.
VNC	Native or virtual running OS accessed with RealVNC or TightVNC.
KVM	KVM is actually design as an accellerator for QEMU, and thus part of the QEMU plugin.
QEMU	With VNC, X11, or CLI modules as remote console. Additionally the SDL console is supported.
VBOX	VirtualBox is going to be integrated within next version.
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

-t one type of session to be actually performed

-T list of session types to be preloaded, e.g. as supporting sub-features, 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.

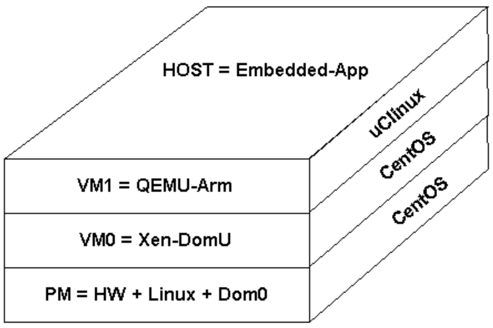


Figure 8.2: Pane-View: QEMU-ARM in Xen-DomU

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.

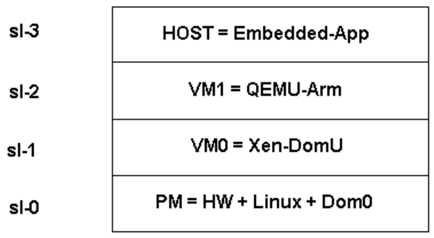


Figure 8.3: Stack-View: QEMU-ARM in Xen-DomU

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.

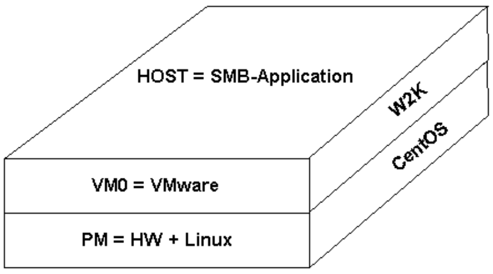


Figure 8.4: Pane-View: W2K in VMware on Linux

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

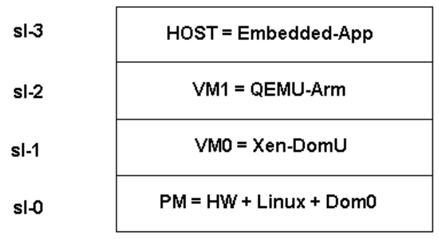


Figure 8.5: Stack-View: W2K in VMware on Linux

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.

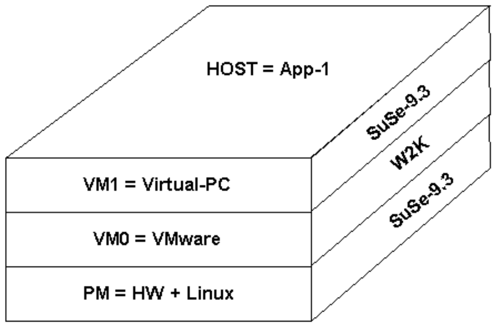


Figure 8.6: Pane-View: Virtual PC with Linux in VMware

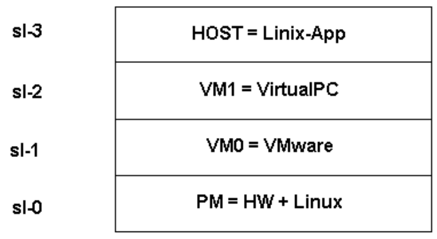


Figure 8.7: Stack-View: Virtual PC with Linux in VMware

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 services 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.

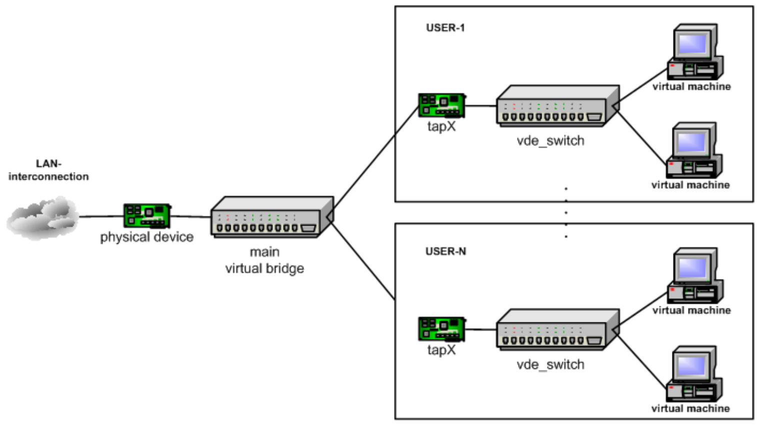


Figure 8.8: Virtual interconnection structure

The implementation of this structure is described in Section **??** '**??**' on page **??** and is automated by the support-tool ctys-setupVDE. 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.

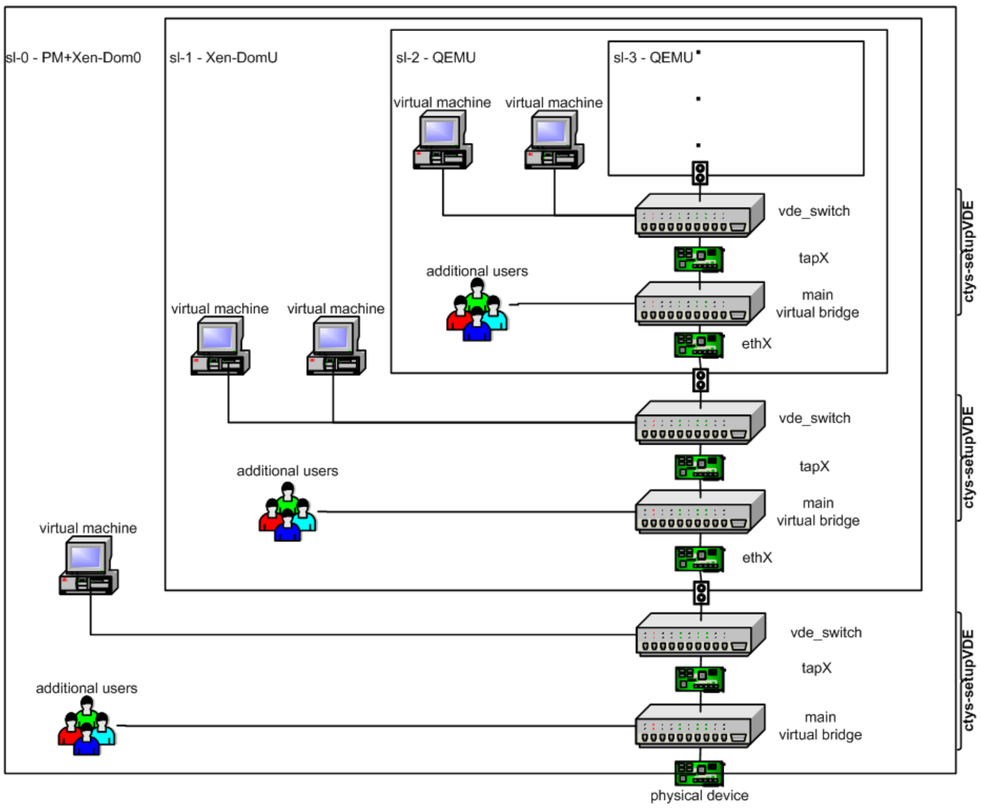


Figure 8.9: Nested Protocol Stacks

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 117 .

The execution flow, as depicted in Figure 11.1 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 11.2.

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.

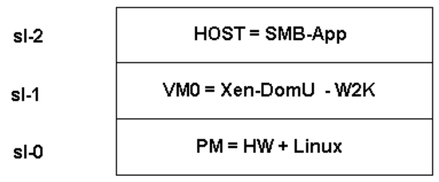


Figure 8.10: Stack-View: W2K as HVM in DomU

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 117 . 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 VMSTACK 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 pre-requires partially some specific resources to be setup and available. Therefore within the the UnifiedSessionsManager the interconnection-attributes

- STACKCAP
  Offered stack capability
- 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:

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

<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 reqularly 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 **??** '**??**' on page **??**), 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 **??** '**??**' on page **??** .

Chapter 9

CTYS-Nameservices

9.1 Basics

The operations of the UnifiedSessionsManager supports a set of tools for nameservices for handling of combined inventory data for physical and virtual machines. The information of virtual machines contains particularly information about offline machines such as installed OS and IP addresses. Additionally the containment information for network based installations is registered in a cache database, where a single VM could be accessed on multiple execution nodes via NFS. The location and relocation capability information is cached within an inventory database and could be utilized by the ctys framework. Severeal views are supported by usage of the common option "-p <database>" representing a specific node-collection. The inventory data is collected and stored for distributed network hosts by the tools

- ctys-vdbgen
- ctys-extractARPlst
- ctys-extractMAClst

without any required user interaction. The store and export format is a ASC-II based record format compatible to OpenOffice and MS-Excel(TM), and any Database. It suits well up to several thousands entries with query-access times in the range of 0.6-0.8 seconds on medium sized machines.

REMARK:
The caching of nameservice data ("-c") data is not related to the caching of payload data for distributed operations ("-C") .

The tools are internally based on the ENUMERATE action of ctys and have the knowledge of the configuration information formats like conf-files of the implemented plugins.

The whole set of tools related to the nameservices of the UnifiedSessionsManager comprise:

- ctys-dnsutil
- ctys-extractARPlst
- ctys-extractMAClst
- ctys-groups
- ctys-genmconf

- ctys-macmap
- ctys-smbutil
- ctys-vdbgen
- ctys-vping

9.2 Runtime Components

The components of the nameservice are structured as depicted within Figure:9.1 on page:81 based on the ENUMERATE and LIST action.

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 optional collection of the distributed runtime information into the inventory database with the runtime tools by themself simplifies the runtime data management by the user and the internal data structures to be implemented significantly.

The data management actions ENUMERATE and LIST provide the keyword MACHINE for output of a raw data format, and the keywords TITLE and TITLEIDX for display of the actual names and canonical indexes of each field. When using the ctys actions as ENUMERATE, the produced output is literally ready to be used within the ctys-nameservice file-database.

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.

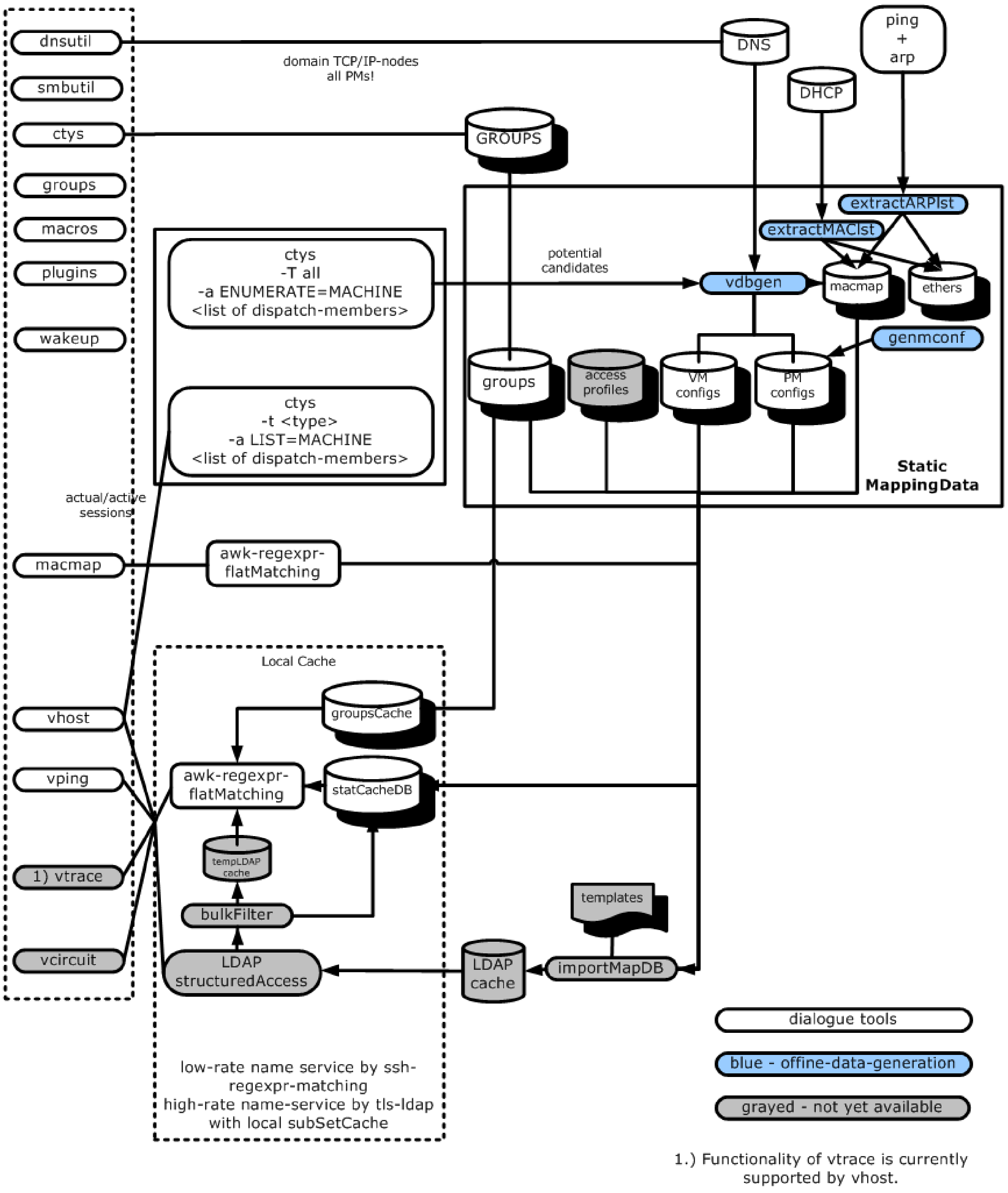


Figure 9.1: Nameservice components

Another 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. 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 network mapping information when required. The individual mapping data is stored 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.

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 pre-combination of groups with each members data is performed.

The components are as given in Figure 9.2. The data could be cached in local databases on

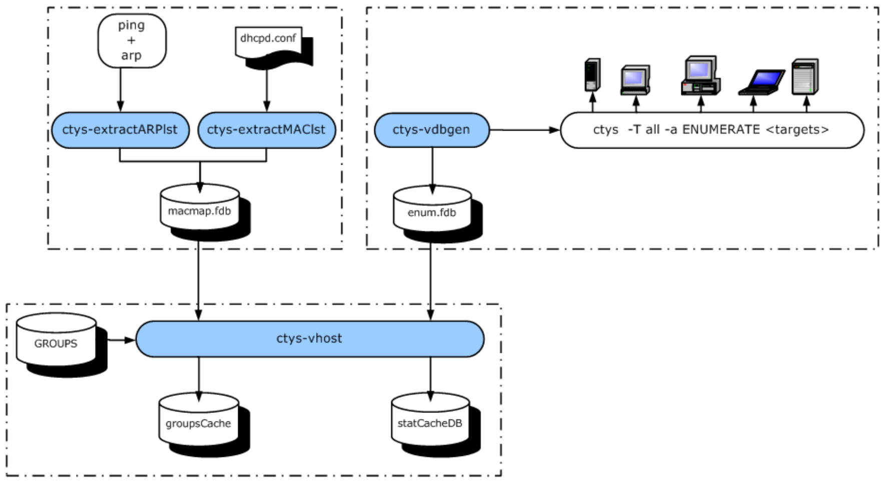


Figure 9.2: Cache Generation

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.

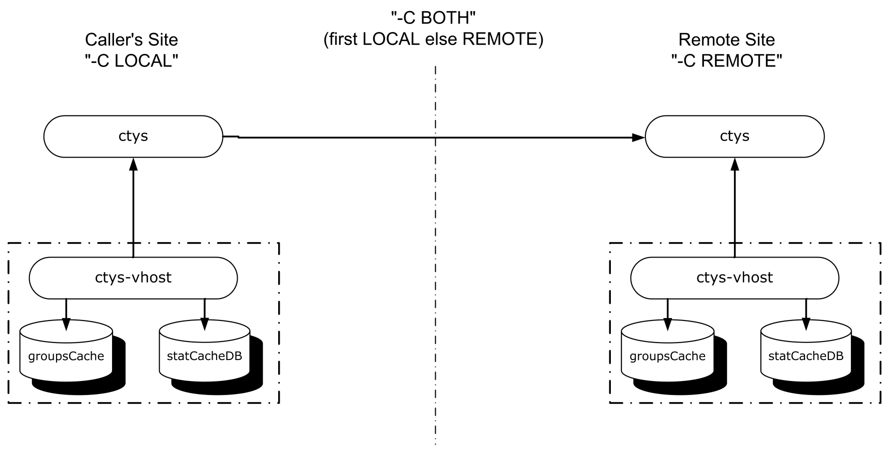


Figure 9.3: Distributed Caches

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

9.2.2 Network LDAP-Access

Even though the performance of the tools seem to be perfectly allright, 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. plugins 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.

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.

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. Hosts could be written either one by line, or as multiple comma seperated entries on one line. 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 "$HOME/.ctys/groups" and "$MYCONFPATH/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.

Goups could be written as a relative pathname to one of the CTYS_GROUPS_PATH entries, allowing the definition of categories of groups.

```
ctys -a list user@desktops/myOfficeSet root@admin/cluster01/basicView
```

For additional information on name resolution for groups refer to "GroupResolution" .

9.5 Addressing Nested Stacks

The operations of the UnifiedSessionsManager utilize nested VMs as stacks of VMs. This includes the implizit start of containing VMs as a runtime base for upper VMs. Thus the startup of an entity within a stack requires some sophisticated information of the pre-required stack structure to be utilized.

The following figure depicts the various configuration entries for an example with 3 Stack-Layers.

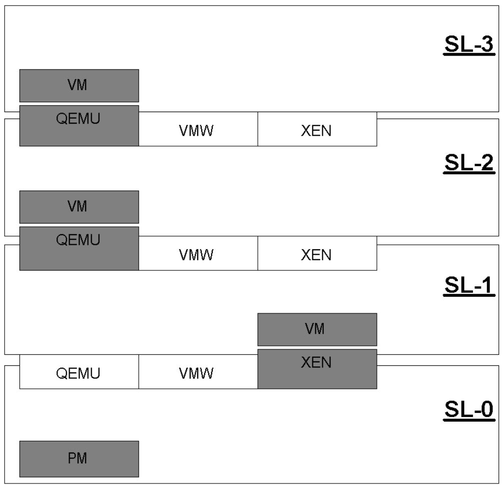


Figure 9.4: Stack-Controller Data

The first layer SL-0 is defined as the lowest layer of the VM-Stack, representing the founding physical machine PM. 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 hypevisor interface for the containig system, the virtual machine attributes for encapsulated guest system. The operation of automatic startup of complete multi-level VM-Stacks requires some specific checks for basic consistency, e.g. the compatibility of the processor architecture. Therefore some extended offline nameservice information is required.

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.

The following figure expands the previous example of Figure: 9.4 with the actual state dependant visibility of the required static configuration data.

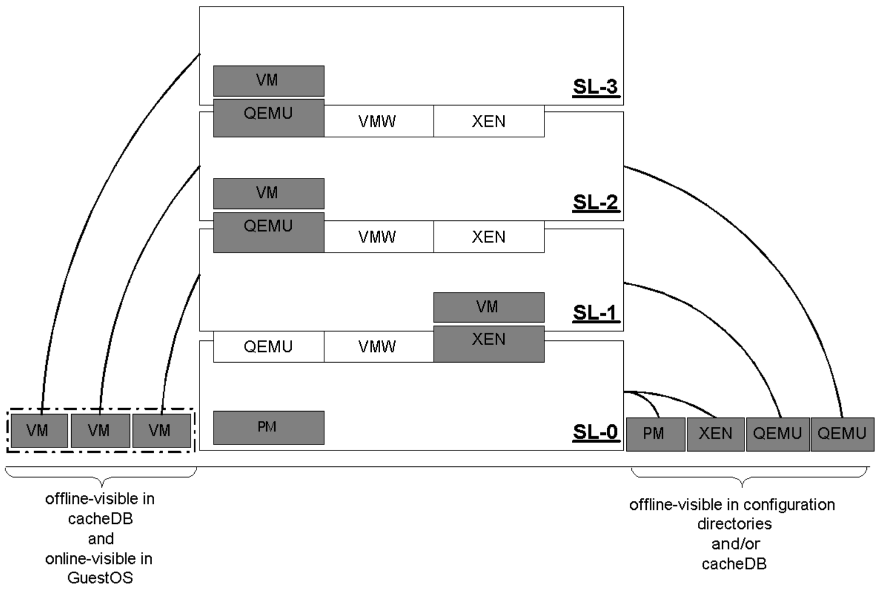

SL-3
VM
QEMU
VMW
XEN
SL-2
VM
QEMU
VMW
XEN
SL-1
VM
QEMU
VMW
XEN
SL-0
PM
VM
VM
VM
PM
XEN
QEMU
QEMU
offline-visible in cacheDB and online-visible in GuestOS
offline-visible in configuration directories and/or cacheDB

Figure 9.5: Stack-Controller Data Visibility

Therefore the UnifiedSessionsManager implements various nameservice measures in order to implement the required features.

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

Part II

Software Design

Chapter 10

Software Architecture

10.1 Hypervisor Sessions Model

The access model to hypervisors within ctys is modelled in a Client-Server fashion. The idea is the representation of a generic service model with a single interface to all service access points.

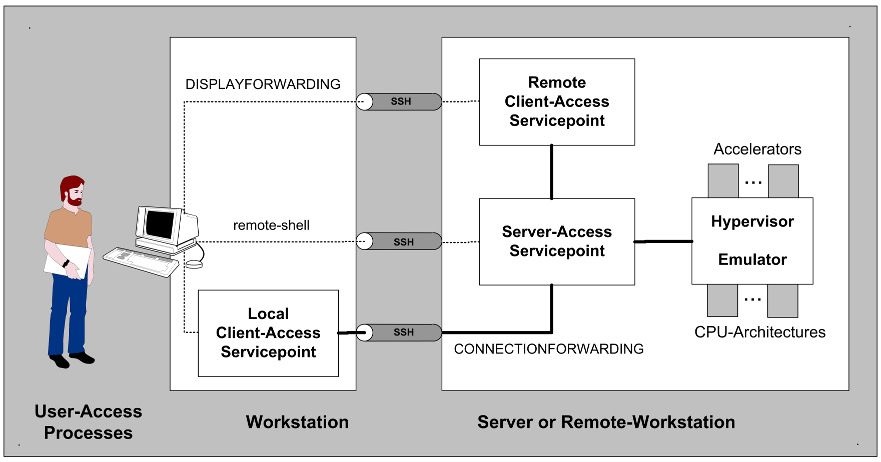


Figure 10.1: Hypervisor Sessions Model

The basic concept herby is the introduction of an abstract service layer in a plugins-bus manner, where each of the various systems is represented by a single unified syntax. Thus all hypervisors are modeled by their expected actual core functionality Hypervisor or Emulator, representing a virtual environment for physical devices. The main application criterias of the virtual hardware for the user herby are the represented CPU-Architectures and the ability to utilize specific accelerators for performance enhancements. These two aspects are particularly represented as attributes for selection and management of access to services by the user.

The given definiton of an abstract service model as depicted within Figure:10.1 founds not just the base for modelling a common acces interface, but the base for facilities to use complete virtual machines as a "simple" virtual software component in virtualized software systems.

The represented basic "Hypervisor Sessions Model" founds the concepts for architectures, where the classical multilayer software design concepts could be seamless virtualized and combined to a set of virtual machines running theirself within one virtual machine - a nested stack of VMs. This completely encapsulates the groups of virtual services within the base

VM as single service access point controlled and presented by standard network access facilities.

Thus the sessions model not only founds the modularization of virtual components, but represents a service architecture which could be customized and maintained in a non-stop manner by widely available standard TCP/IP-facilities.

10.2 Basic Modular Design

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

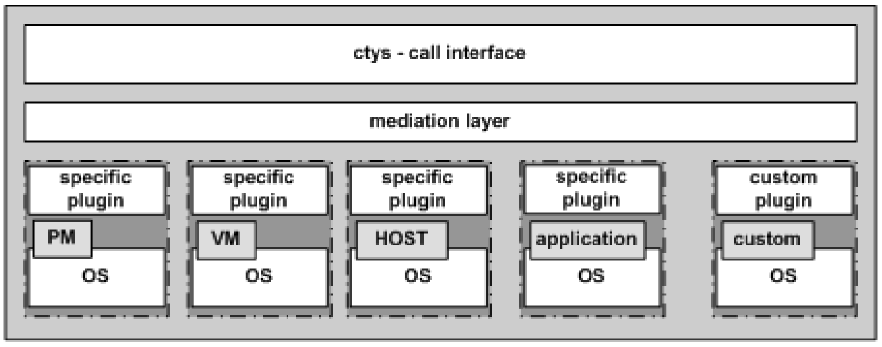


Figure 10.2: ctys Software Layers

The implementation of the plugin model by exchangeable components is similar to "Shared Objects" or "Shared Libraries" and based on the "source" function of bash. The detection and integration is based on dynamic scan of subdirectories in combination with a load mechanism and operational state management. The usage of the bash has some limitations, but offers simplicity and common applicability for the first version.

The execution of ctys is splitted into a client and a server component, where both components are identical, but perform different functions in their specific execution context.

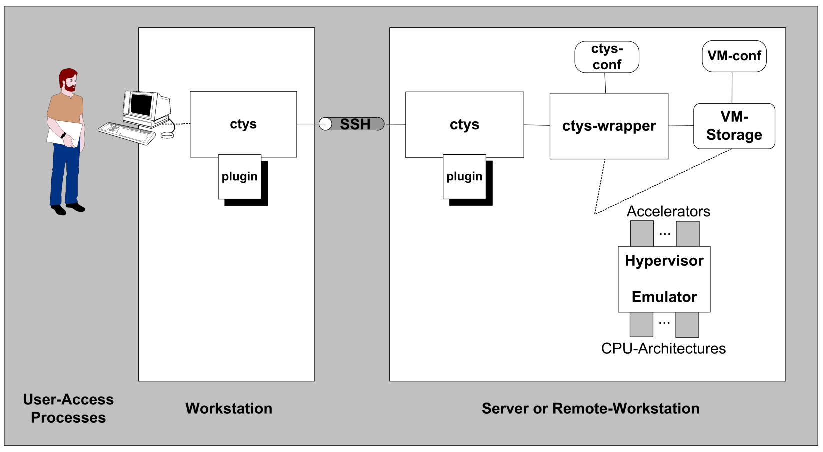


Figure 10.3: ctys distributed components

The distributed components rely on the basic structure of a common framework with generic services such as name services, task and sessions management, and distributed data services for inventory management. The main feature of the generic approach is the common distributed applicability of framework functions as well as framework executables from the standard user interface.

The provided interfaces for runtime-integration supports in-process-interfaces for plugins and inter-process interfaces for framework tools and wrapper calls.

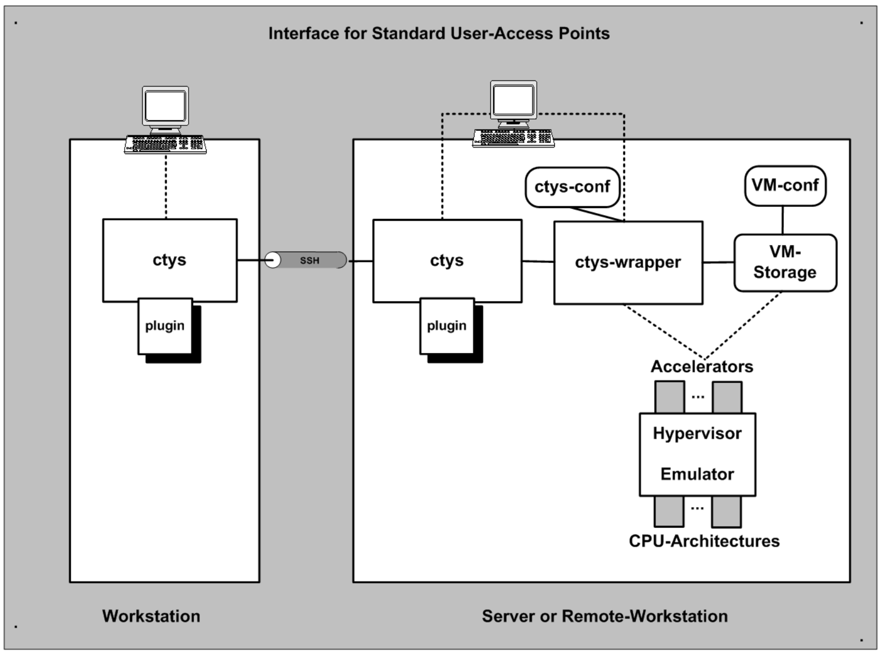


Figure 10.4: ctys distributed components

Thus an integrated custom application is capable for distributed operations by default, where a basic version management for distributed compatibility issues is integrated.

10.3 Communications Model

The communications between entities within the UnifiedSessionsManager is modeled around the basic idea of transparent access to user desktops for local and remote logins with extensions for pure command line sessions. This results in IO-stream based communications, which is limited but quite simple to implement.

The consequence of this approach is a resulting 2-type category of communications. The first is the forwarding of the display to the user for remote actions, either complete remote desktops or simple remote shells as the managing entity for user sessions. The second is the local execution of the user interface for the management of current session, where the local sessions client opens communications to remote services.

The resulting architecture is a layered architecture modeled as an independent communications service layer and a sessions management layer which is basicly peer-to-peer oriented only. Thus the communications is handeled on high level abstraction with an addressing abstraction called <machine-address>. The remaining knowledge of communications type for the sessions layer are the two types as basic pattern

DISPLAYFORWARDING

The whole application load including the access facilities such as the user desktop application is located on the server site. Local resources are required for basic screen display only. This mode suits particularly for thin clients.

CONNECTIONFORWARDING
The application server is located on the server, whereas the client services are based on the local machine. This includes the access such as a desktop application as well as specific client processing services of the application. This mode suits particularly for applications with increased autonomous client side processing with less communications requirements to server based services. In case of distributed clients with local data caches this might be the best approach.

10.4 Security Model

The main aspect for the UnifiedSessionsManager as a sessions management application is the protection of the inter node communications. Therefore the approach is similar to the basic idea of Kerberos, where for local security on the communications peers some additional measurements of the platform facilities are required.

By default a basic user based access and view model for local access is implemented, which relies on the access permissions of the local user account.

The communications supported are based on encryption only, in current version on OpenSSH. Therefore the authentication and authorisation could be varied by the means of Single-Sign-On provided by TLS/GSSAPI.

Chapter 11

Runtime Interfaces

11.1 Target-Platforms

The support for multiple platforms requiers specific dynamic adaptions for each, which are based within the ctys framework mainly on the variables MYOS and MYDIST. These contain configuration entries, path dependencies, platform specific option sets of system utilities and in some cases specific functionality for adapted processing.

The convention is based on in-source control flow as well as specific modules, where the naming convention e.g. "qemu-$MYOS.conf" is applied. One example for this is the distinction between Solaris and OpenSolaris, here both share the same MYOS, but distinguish in MYDIST. And bot together require common specific, but additionally distinguish theirselfs by some minor differences. The same for the MYOS=Linux, where MYDIST widely varies e.g. to CentOS, debian, SUSE, Ubuntu, etc. In some cases additional aspects such as the CPU architecture are required to be applied.

A set of common generic components is included but may fail partly due to minor differences when not adapted. The most of current distributions are supported by appropriate adaptions.

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

DISPLAYFORWARDING

This is the default mode for usage of 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 images 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 is used. This is 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 images 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. When using CONNECTIONFORWARDING the things become some more complicated. The decision here was made to using port forwarding 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.

11.3 Control and Data Flow

The basic design concepts for the current implementation of ctys comprise context based relocateable services, the single-type-of-access services for local and network based access with the combination of facilities for a distributed controller for task data and execution. Additionnally the design for the feature set is expanded into a concurrent execution set based on nested execution sets.

11.3.1 Distributed Controller

The design of the serverless execution as a relocatable distributed controller suits particularly for the current implementation where the whole design is mainly based on volatile dynamic runtime data only. The caching of distributed data into an inventory database is provided a an optional feature for performance enhancement and enhanced features. Thus each networked function requires frequently the synchrounous and asynchronous propagation of services to one or more remote locations and the final postprocessing of the combined results.

The design of this functionality reuiqres not only aspects of job control and sessionsmanagement, but the decision of splitting service execution and the distribution of task loads too. This particularly requires some effort for the design of the application of user defined options within one chain of a distributed workflow.

Therefore some types of local, remote, and mixed applicable options with a base facility for locating and forwarding diverse subsets of call options is designed within the Distributed Contoller.

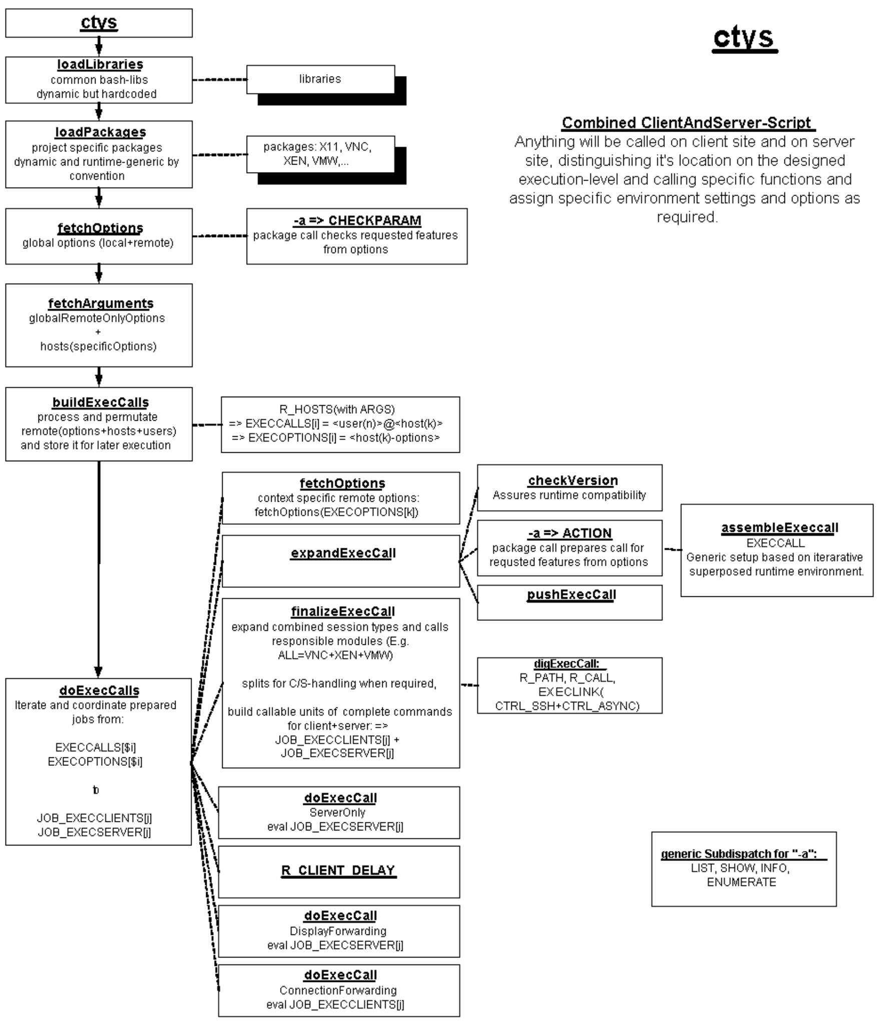

ctys
Combined ClientAndServer-Script
Anything will be called on client site and on server site, distinguishing it's location on the designed execution-level and calling specific functions and assign specific environment settings and options as required.
ctys
loadLibraries
common bash-libs
dynamic but hardcoded
libraries
loadPackages
project specific packages
dynamic and runtime-generic by convention
packages: X11, VNC, XEN, VMW,...
fetchOptions
global options (local+remote)
-a => CHECKPARAM
package call checks requested features from options
fetchArguments
globalRemoteOnlyOptions
+
hosts(specificOptions)
buildExecCalls
process and permutate remote(options+hosts+users) and store it for later execution
R_HOSTS(with ARGS)
=> EXECCALLS[i] = <user(n)>@<host(k)>
=> EXECOPTIONS[i] = <host(k)-options>
fetchOptions
context specific remote options:
fetchOptions(EXECOPTIONS[k])
expandExecCall
checkVersion
Assures runtime compatibility
-a => ACTION
package call prepares call for requsted features from options
pushExecCall
assembleExeccall
EXECCALL
Generic setup based on iterarative superposed runtime environment.
finalizeExecCall
expand combined session types and calls responsible modules (E.g. ALL=VNC+XEN+VMW)
splits for C/S-handling when required,
build callable units of complete commands for client+server: =>
JOB_EXECCLIENTS[j] +
JOB_EXECSERVER[j]
digExecCall:
R_PATH, R_CALL,
EXECLINK(
CTRL_SSH+CTRL_ASYNC)
doExecCalls
Iterate and coordinate prepared jobs from:
EXECCALLS[$i]
EXECOPTIONS[$i]
to
JOB_EXECCLIENTS[j]
JOB_EXECSERVER[j]
doExecCall
ServerOnly
eval JOB_EXECSERVER[j]
R_CLIENT_DELAY
doExecCall
DisplayForwarding
eval JOB_EXECSERVER[j]
doExecCall
ConnectionForwarding
eval JOB_EXECCLIENTS[j]
generic Subdispatch for "-a":
LIST, SHOW, INFO,
ENUMERATE

Figure 11.1: ctys Local Control Flow

11.3.2 Task Data

The task data as defined here contains multiple aspects as defined by ctys. The main aspect is the handling of multiple execution targets including - later - tree-style distribution of subtasks either parallel, sychrounous, asynchronous and with specific option sets in completely mixed and heterogeneous excution set. The distribution of jobs and reuiqred handling of specific option sets is particularly supported by the GROUP and MACRO features of ctys.

The integrated management of multi-screen and multi-workspace desktop environments requires some tainting of the pure task based job and session control. This is due to practical aspects of pop-up windos on several layers of so called workspaces, where the so called "z-axis" for the GUI allows only the topmost to be visible to the user. Thus some reordering of tasks with visual representation elements on the desktop has to be performed. This restriction applies to ctys due to it's serverless, thus basically stateless operations.

Additionally the requirements for execution-options-allocation for distributed tasks apply.

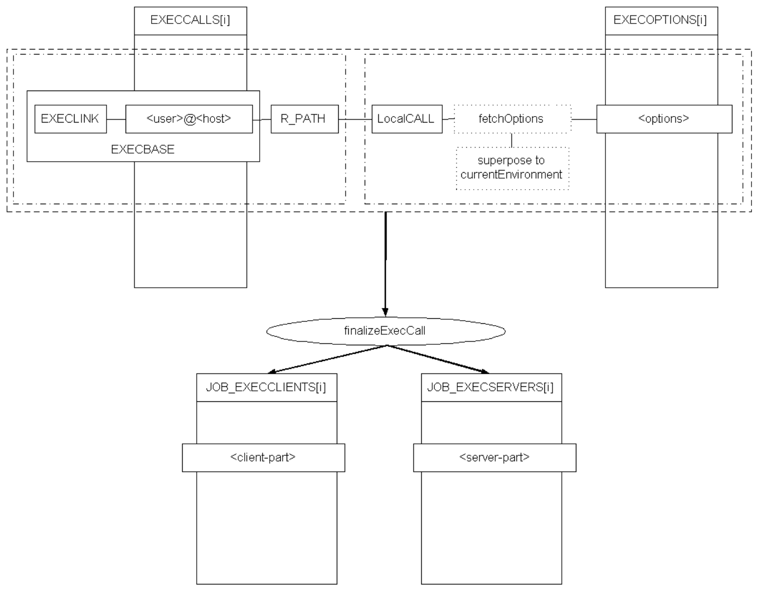


Figure 11.2: Task Data handled by the main dispatcher

The Figure: 11.2 depicts some basic facilities for implementing the handling of execution attributes - here command line options.

11.3.3 Stack Interworking

The combination of complete virtual machines into nested stacks requires particularly some effort related to handling the runtime states and their propagation within encapsulated submachines. Thereofre the concept of integrated Stack-Interworking is implemented which comprises then physical machine - PM - as well as the whole heteregeneous nested stack of virtual machines - VMs - executed on the PM.

The main basic aspects are the automation of startup and shutdown of intermediate stack entities, where multiple contained services could be affected. Thus mechanisms for implicit bottom-up startup of stack-prerequirements as well as top-down shutdown of dependent superservices is designed and implemented. The basic idea is here the propagation of runtime states initiating appropriate actions within each entity of the propagation-chain.

Create Propagation - CREATE

To be documented.

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.

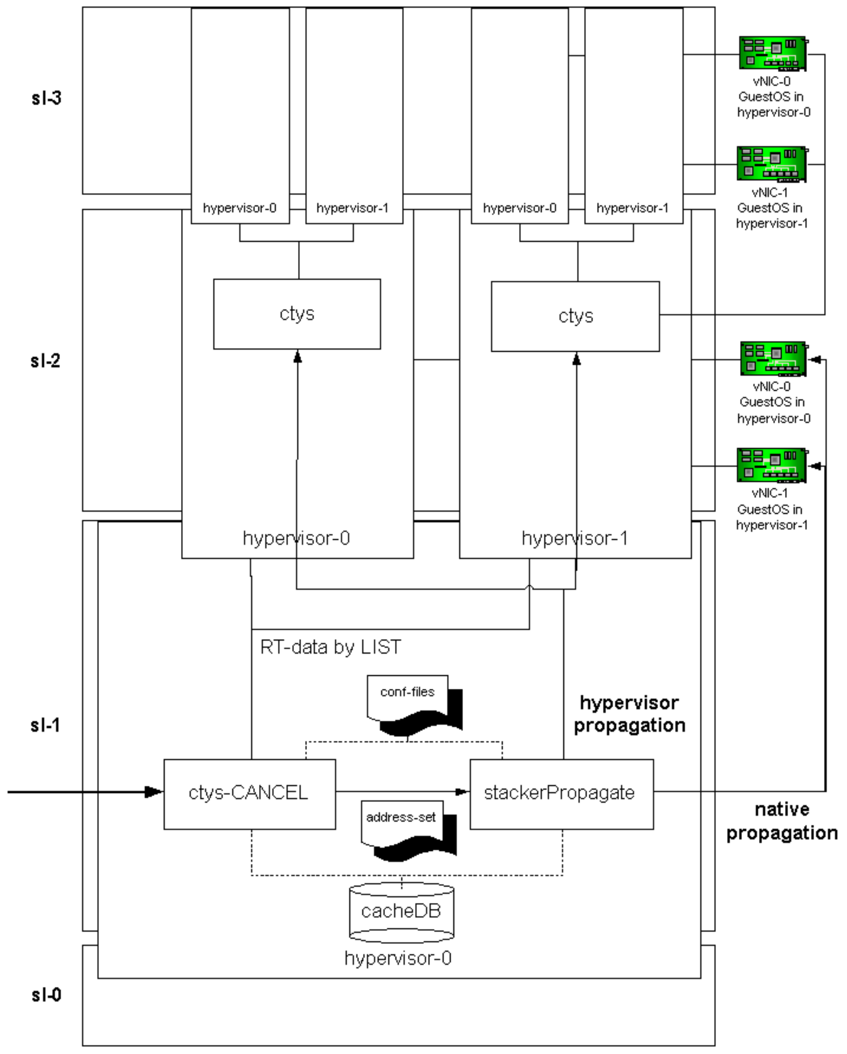


Figure 11.3: Nested Upward-Stackpropagation

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 timeout 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 VMwareTools.

   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

To be documented.

11.4 Plugins Integration

11.4.1 Basics on "bash"

The benefit of the bash is it's applicability due to commonly available knowledge of the targeted users. The usability of ctys including the framework functions within shell scripts, offers a distributed processing facilities including the management of remote data and processing. This particularly may compensate the lack of support for complex data structures and within the bash.

The most important feature used within the UnifiedSessionsManager is the "source" of components. This is developed within ctys tools to be used as a similar approach to shared libraries of the binaries execution environment. Therefore almost all scripts of the ctys tools are just plugged together by using a load mechanism based on "hook" convention.

11.4.2 Component Framework

The framework offers particularly a complete set of functions for distributed execution of embedded functions and command line based executables. The embedded functions as loadable bash-plugins could be particularly wrapper for almost any type of subcalls including starters for graphical interfaces.

The most important interface is the concept and implementation of the dynamic and generic access mechanism for loadable plugins. This is based on some minor naming convention and a basic set of interfaces including an init-level based set of initialization calls and propagation of these states to subcalls.

Static Load of Modules

ffs.

Dynamic OnDemand Load of Modules

ffs.

Operational States

The main idea behind the introduction of operational states is the reduction or better elimination of static runtime dependencies by introduction of dynamic adaptable feature sets. This is exactly implemented within ctys.

Therefore the initial startup processing scans for actual available feature sets and decides whether to continue or not. This detection and decision process is splitted into multiple scopes and levels by multiple internal so called distributed state machines. Each component decides within it's current runtime context whether a specific feature is available, and thus the requested service could be offered. The validation tool "ctys-plugins" itself is the best example. The scope of operations could be controller by selecting the actually loaded components as well as defining a runtime context by the "-E" option. This option defines whether the runtime context is "client" or "server", thus requests server or client services either as mandatory or as optional. This generic distributed approach eliminates the issues related to SW-Installation and distribution of specific versions.

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 in the current runtime-context. This allows the self-propagated-installation of ctys by "ctys-distribute" with less prerequisites.

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 state variable for a plugin is dynamically determined based on the array of names for 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, VMW_STATE, QEMU_STATE, or VNC_STATE
```

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

AVAILABLE(0)
This is the implicit state of any found module, which is actually not handled, but just present. Any present plugin is registered in a list of available plugins, which represents it's presence. The default state is set to DISABLED, the first "level" of managed states.

DISABLED(1)
The default state stored in the runtime modules.

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

IDLE(3)
This state is currently not be utilized.

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 special flag similar to DISABLED is available, which influences the loader and prevents the load of the plugin at all. So the IGNORED state is actually the "unavailable" state, avoiding execution failure e.g. during initialization on an unsupported platform. Refer to the configuration file "ctys.conf", a detailed example is provided within Section **??** '**??**' on page **??** .

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

Multi-OS Boot Environments

The usage of multi boot environments opens numerous issues to it's management. One is the installation of appropriate software components and avoiding the of usage of 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 state variables. 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 be usable on a configured kernel for VMware.

11.4.3 Dispatcher

ffs.

11.4.4 Common Data Structures

The interface data depicted within this chapter could be displayed by various formats. The following formats are supported:

1. REC
   A propriatary record format:

   ```
   record(#rec-idx):={
      {#field-idx, attr-name, attr-val},
      {.....
   }
   ```

2. SPEC
   A meta-data record format for testing of data with easy readabilty:

   ```
   record(#rec-idx):={
      #field-idx                attr-name: attr-val
      #field-idx                attr-name: attr-val
      #field-idx                attr-name: attr-val
      .....
   }
   ```

3. TAB
   A table format:

   ```
   |attr-name    | attr-name    | attr-name    |
   +------------+-------------+-------------+
   |attr-val     | attr-val     | attr-val     |
   |attr-val     | attr-val     | attr-val     |
   ...
   ```

4. XML
   An export format for post-processing:

   ```
   <record index=#rec-idx>
      <attr-name index=#field-idx>attr-val</attr-name>
      <attr-name index=#field-idx>attr-val</attr-name>
      .....
   </record>
   ```

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	HYPERRELRUN	Release of actual locally available hypervisor.		29
30	ACCELERATOR	Accelleration component.		30
31	EXEPATH	The FQDN of the executable.		31
32	RESERVED	For internal 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
44	UID	The user ID of current account.		44
45	GID	The group ID of current account.		45
46	defaultHOSTs	The default type of native login.		46
47	defaultCONSOLE	The default type of console for hypervisor.		47

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	HYPERRELRUN	Release of actual locally available hypervisor.		29
30	ACCELERATOR	Accelleration component.		30
31	EXEPATH	The FQDN of the executable.		31
32	RESERVED	For internal 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
44	UID	The user ID of current account.		44
45	GID	The group ID of current account.		45
46	defaultHOSTs	The default type of native login.		46
47	defaultCONSOLE	The default type of console for hypervisor.		47

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

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
16	IFNAME	The name of the interface.		16
17	RESERVED	FFS		17
18	CONTEXTSTRG	Plugin specific context string.		18
19	EXEPATH	The FQDN of the executable.		19
20	HYPERRELRUN	Release of actual running hypervisor.		20
21	ACCELERATOR	Acceleration component.		21
22	ARCH	Provided runtime architecture		22

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
16	IFNAME	The name of the interface.		16
17	RESERVED	FFS		17
18	CONTEXTSTRG	Plugin specific context string.		18
19	EXEPATH	The FQDN of the executable.		19
20	HYPERRELRUN	Release of actual running hypervisor.		20
21	ACCELERATOR	Accelleration component.		21
22	ARCH	Provided runtime architecture		22

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

11.4.5 Categories

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.

11.4.6 bash-Plugins and bash-Libraries

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.

Chapter 12

CTYS-Nameservices

12.1 Runtime Components

The components of the nameservice are structured as depicted within the figure:9.1 on page:81 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:110.

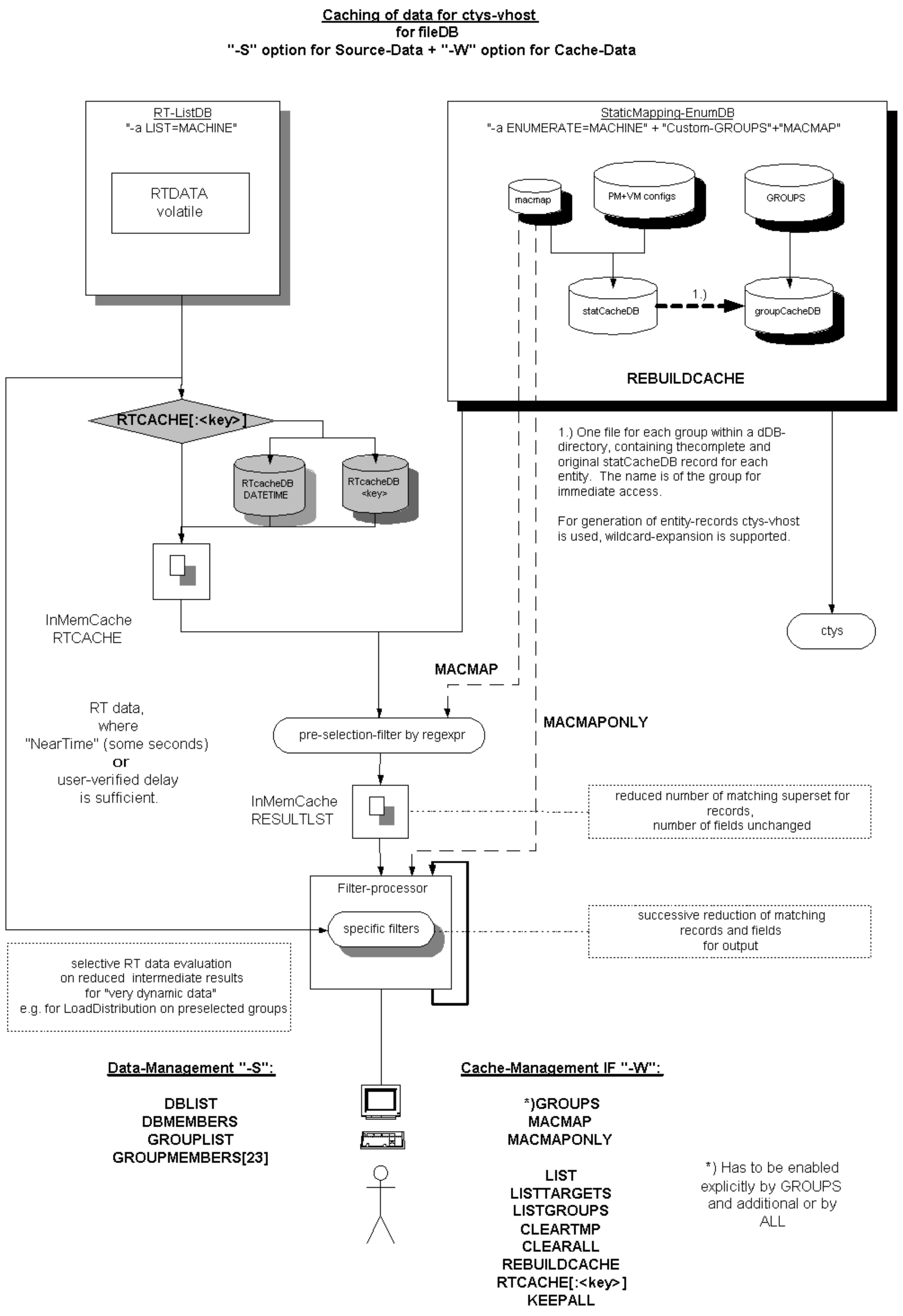

Caching of data for ctys-vhost
for fileDB
"-S" option for Source-Data + "-W" option for Cache-Data
RT-ListDB
"-a LIST=MACHINE"
RTDATA
volatile
StaticMapping-EnumDB
"-a ENUMERATE=MACHINE" + "Custom-GROUPS"+"MACMAP"
macmap
PM+VM configs
GROUPS
statCacheDB
1.)
groupCacheDB
REBUILDCACHE
RTCACHE[:<key>]
RTcacheDB DATETIME
RTcacheDB <key>
1.) One file for each group within a dDB-directory, containing thecomplete and original statCacheDB record for each entity. The name is of the group for immediate access.
For generation of entity-records ctys-vhost is used, wildcard-expansion is supported.
InMemCache
RTCACHE
ctys
MACMAP
RT data, where "NearTime" (some seconds) or user-verified delay is sufficient.
pre-selection-filter by regexpr
MACMAPONLY
InMemCache
RESULTLST
reduced number of matching superset for records, number of fields unchanged
Filter-processor
specific filters
successive reduction of matching records and fields for output
selective RT data evaluation on reduced intermediate results for "very dynamic data" e.g. for LoadDistribution on preselected groups
Data-Management "-S":
DBLIST
DBMEMBERS
GROUPLIST
GROUPMEMBERS[23]
Cache-Management IF "-W":
*)GROUPS
MACMAP
MACMAPONLY
LIST
LISTTARGETS
LISTGROUPS
CLEARTMP
CLEARALL
REBUILDCACHE
RTCACHE[:<key>]
KEEPALL
*) Has to be enabled explicitly by GROUPS and additional or by ALL

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>
```

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

Arguments <execution-targets> 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.

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.

This changes, 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 group resolution and the accessibility of several desktops, some reordering and grouping of tasks can appear.

Pre-required 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.

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

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 by simply nesting loops and using sets generated from basic regular expressions. This is particularly important for simplification of custom plugins. 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 multiple execution-targets as combined group entity. A group entity is a logical unit with it's own execution context.

Group objects are mapped within ctys to one or more specific sub-processes, 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.

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 members of a hierarchical stack, thus sequentially dependent on each other from-left-to-right.
- VCIRCUIT
  Sets up an sequential relay chain with pre-assigned intermediate nodes for establishing an double-encrypted tunnel. The bypassing of Firewalls for specific access groups is typical application.

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.

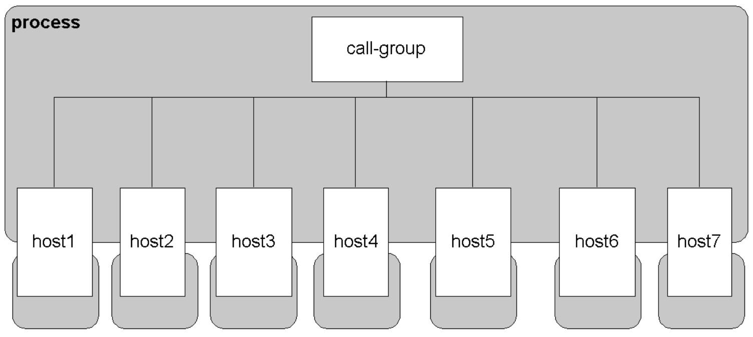


Figure 13.2: Groupresolution by Include only

The group object can replace any valid execution target and supports context options. 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)'...
```

Any level of nested includes is supported. circular inclusion will be detected at an default level of 20 and terminated than.

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

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)'  \
```

For current version nesting of braces is NOT supported, but chaining of braces IS. Permutation is performed for now only for the first level of group resolution. Specific points ofr current scanner 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 within the bash.

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:115.

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.

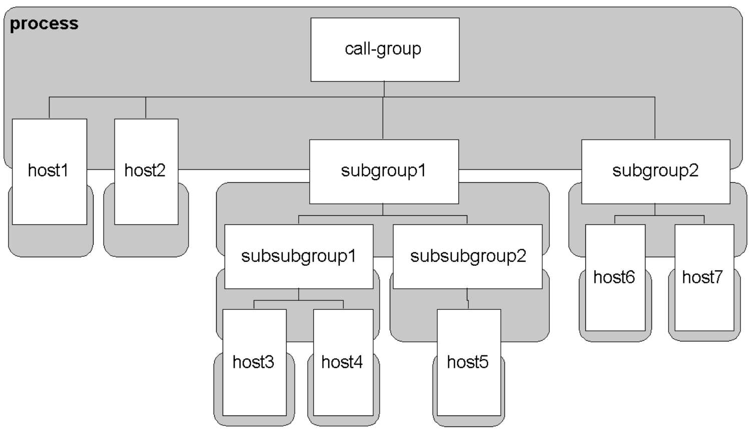


Figure 13.3: Groupresolution by Subgroups

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 various VMs on the same PM.

13.3.4 Stacks as Vertical-Subgroups

The VM-Stack implementation by it's design close to a Sub-Group with just a fixed set of context options.

The seamless extension of the design concept of SUBGROUPs to VMSTACKs, extends the "flat-feature" and "horizontal-feature" . These concepts could be applied intermixed, but the nesting of VMSTACKS is not provided.

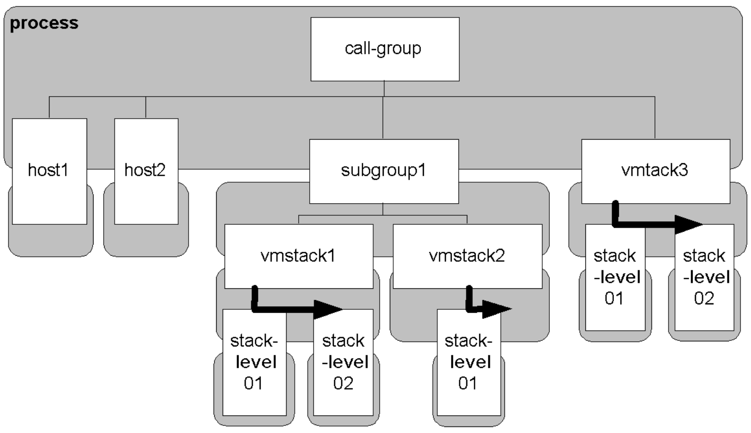


Figure 13.4: Combined Subgroups and Substacks

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.

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 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.

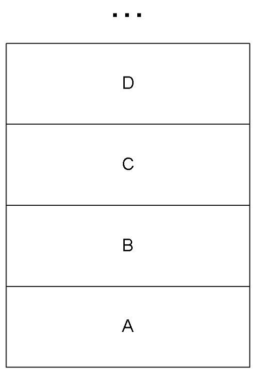


Figure 13.5: Stack Example for Basic Call-Interface

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) is assured by sequential left-to-right operation of a VMSTACK.

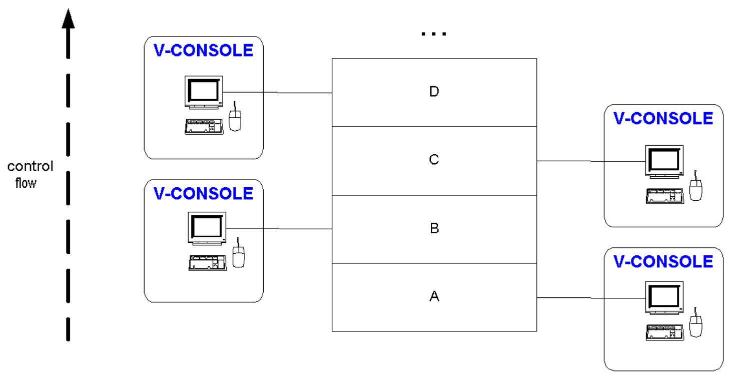


Figure 13.6: CONSOLE- and HOSTs-Asynchronity for Stacked-Execution

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 incrementally startups the stack on the actual physical machine "A", after it's activation by usage of the relay wolExecRelayServer(refer to Section **??** '**??**' on page **??**).

```
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-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=l: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.

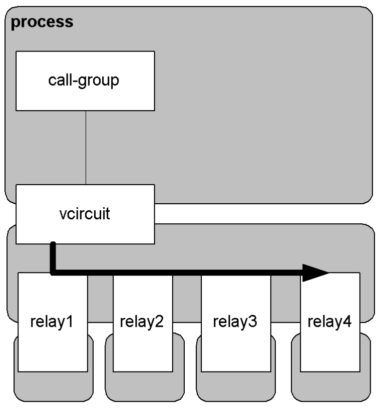


Figure 13.7: VCIRCUIT

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.

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 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".

(CHDIR|CD):<working-directory-change-to> Change current working directory on remote site before execution of the remote access. This is currently applicable for the X11 and CLI plugins only.

CMD:<cmd> Replaces the standard definition of a command execution shell by "CLI_SHELL_CMD_DEFAULT='bash -c'" for remote execution.

<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 **??** '**??**' on page **??** .

Chapter 14

Core Data

14.1 Overview

The internal static configuration data is based on the output records of the ENUMERATE action which has a key role in 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 required functionality for off-line PMs and VMs as well, as a reduction of the average query-time to less than a second. 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.

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 bld/out/doc-tmp/en/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 seamless 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/KVM [1]	ctys
VBOX	ctys
VMW	vmx,ctys
XEN [2]	conf,ctys

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.

ACCELERATOR
: The accellerator available for ENUMERATE or actually executed for LIST. The available types of accellerator depends on the session type. These are for example:

QEMU : KQEMU, KVM, QEMU

VBOX : PARA, HVM

VMW : PARA, HVM

XEN : PARA, HVM

ARCH
: Virtual architecture presented to the GuetsOS and hypervisors of upper-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

[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.

organization, as well as basis 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.

EXEPATH
The actual path of the executable service access entity.

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.

HYPERRELRUN|HRELRUN|HRELX|HRX
: The release of the current hypervisor. This is in case of ENUMERATE the locally available and foreseen hypervisor, whereas in case of LIST this is the actually executed hypervisor, which in case of some plugings could be varied by call parameters.

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 **??** '**??**' on page **??**).

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.

DEACTIVATED
The VM is present but the configuration data could only be accessed partially. The common reason is the missing of access permissions, in some cases a registration of the VM to the native administration component is required. The concrete definition and required actions are defined by the specific plugins.

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%...".

REC_GEN|REC:<tab-args>
: Defines proprietary records for formatted output.

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".

SPEC_GEN|SPEC:<tab-args>
Defines check records for formatted output.

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 GSSAPI/TLS, thus the <credentials> field is not yet supported.

XML_GEN|XML:<tab-args>
Defines XML records for postprocessing output.

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 INFO, LIST, ENUMERATE, and SHOW 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 **??** '**??**' on page **??** .

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.

14.7 Generic Records

The GENERIC class of actions LIST and ENUMERATE, additionally 'ctys-vhost' support data to be displayed in record formats.

The output could be adapted by the user with the same suboptions as for generic tables, but representing a line oriented attribute output.

The following formats are currently available:

1. REC
   A propriatary record format:

   ```
   record(#rec-idx):={
      {#field-idx, attr-name, attr-val},
      {.....
   }
   ```

2. SPEC
   A meta-data record format for testing of data with easy readabilty:

   ```
   record(#rec-idx):={
      #field-idx               attr-name: attr-val
      #field-idx               attr-name: attr-val
      #field-idx               attr-name: attr-val
      .....
   }
   ```

3. XML
   An export format for post-processing:

   ```
   <record index=#rec-idx>
      <attr-name index=#field-idx>attr-val</attr-name>
   ```

```
    <attr-name index=#field-idx>attr-val</attr-name>
    .....
</record>
```

.

Chapter 15

Address Syntax

This document describes the common generic address syntax for single machines and groups of entities. This suffices all supported systems and may for some plugins applicable as a subset only.
The current version provides almost only the <**machine-address**> and the **GROUPS** objects, thus the remaining definitions were required for the design of an extendable overal concept.

15.1 Basic Elements

The addressing facility including the namebinding is splitted into a logical description as a general view and it's conrete adaptions which could be implemented by multiple presentations. The foreseen and implemented syntax scanners are designed to allow implementation in a straight-forward manner allowing an simple implementation of hierarchical structured syntax definitions by nested loops.

The following characters are reservered syntax elements, the full set and description is given in the chapter "Options Scanners - Reserved Characters".

'=' Seperator for option and it's suboptions.

',' Seperator for suboptions belonging to one set of suboptions.

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

The current syntax description may not yet formally be absolutely correct nor complete, but may cover the intended grade of open description and required understanding for it's application. Some modifications are still under development.

15.2 SyntaxElements

The following namebinding founds the superset of addressing attributes, which supports explicit addressing of targets as well as generic addressing of multiple targets by using search paths and content attributes in analogy to wildcards, 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.

The top-level addressed entity is the APPLICATION, thus here the <**target-application-entity**>. This contains in analogy to the OSI model the machine as well as the access port.

```
<target-application-entity>:=<tae>
<tae>:=[<access-point>]<application>

<access-point>:={
   <physical-access-point>
   |<virtual-access-point>
   |<physical-access-point>[<virtual-access-point>]
   |<group-access-points>
}

<physical-access-point>:=<pm>
<pm>:=<machine-address>[:<access-port>]

<virtual-access-point>:='['<vm>']'
<vm>:=<machine-address>[:<access-port>]

<group-access-points>:=<group>[:<access-port>]

<application>:=<host-execution-frame><application-entity>
```

Figure 15.1: TAE - Target Application Entity address

The machine is addressed by the <**machine-address**>, which represents physical and virtual machines as well as login-sessions provided by the HOSTs plugin. The specific plugins may suppport a subset of the full scope, but the attributes **ID** and **LABEL** are mandatory in any case. The **ID** attribuet is here either a persistent identifier, in case of a VM a configuration file, or a dynamic identifier in case of the HOSTs plugin, e.g. for VNC the DISPLAY number excluding the port-offset. Whereas it is defined for X11 as the PID.

```
<machine-address>:=
  (
     [(ID|I|PATHNAME|PNAME|P):<mconf-filename-path>][,]
     |
     [(ID|I):<id>][,]
  )
  [(BASEPATH|BASE|B):<base-path>[\%<basepath>]{0,n}
  [(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

The GROUPS objects are a concatination of <machine-addresses> and nested GROUPS including specific context options.

```
<group-address>:= (
          [ <machine-addresses>['(' <machine-options> ')']{0,n}]
          [ <group-address>['('     <group-options>   ')']{0,n}]
)['('<group-options>')']
```

Figure 15.3: Group-Address

The <**DISPLAYext**> addresses a network display, where the full bath includes the <**target-application-entity**>, thus providing for various addressing schemas including application gateways.

```
<DISPLAYext>:=<target-display-entity>

<target-display-entity>:=<tde>
<tde>:=<tae>:<local-display-entity>
<local-display-entity>:=<lde>
<lde>:=(<display>|<label>)[.<screen>]
```

Figure 15.4: 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 15.5: 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>:<lde>
```

(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 pre-required - 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 117), 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˜15.6.

```
<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 15.6: 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 operation 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 Resolution

Groups are valid replacements of any addressed object, such as a HOST. Groups can contain in addition to a simple set of hostnames a list of entities with context specific parameters and include other groups in a nested manner. Each set of superposed options is permutated with the new included set.

The resolution of group names is processed by a search path algorithm based on the variable

,

CTYS_GROUPS_PATH , which has the same syntax as the PATH variable. The search algorithm is a first-wins filename match of a preconfigured set. Nested includes are resolved with a first-win algorithm beginning at the current position.

In addition to simple names a relative pathname for a group file could be used. This allows for example the definition of arbitrary categories, such as server, client, desktop, db, and scan. Here are some examples for free definitions of categories based on simple subdirectories to search paths. The level of structuring into subdirectories is not limited.

server/* A list of single servers with stored specific call parameters. Server is used here as a synonym for a backend process. Which could be either a PM or a VM, the characteristics is the inclusion of the backend process only.

client/* A list of single clients with stored specific call parameters. This is meant as the user front end only, which could be a CONNECTIONFORWARDING. The user can define this category also as a complete client machine including the backend and frontend, which is a complete client for a service.

desktop/* A composition of combined clients and servers for specific tasks. This could be specific desktops for office-applications, systems administration, software-development, industrial applications, test environments. Either new entries could be created, or existing groups could be combined by inclusion.

db/* Multiple sets of lists of targets to be scanned into specific caching databases. This could be used for a working set as well as for different views of sets of machines.

scan/* A list of items to be scanned by tools for access validation and check of operational mode. Therefore this entities should contain basic parameters onyl, such as machine specific remote access permissions type.

REMARK: The group feature requires a configured SSO, either by SSH-Keys of Kerberos when the parallel or async mode is choosen, which is the default mode. This is required due to the parallel-intermixed password request, which fails frequently.

For additional information on groups refer to "GroupTargets" and "ctys-groups" .

15.5 Groups of Machines

The GROUPS objects are a concatination of <machine-addresses> and nested GROUPS including specific context options. The end of the command with it's specific option should be marked by the common standard with a double column '–'.

```
ctys -a <action> -- '(<glob-opts>)' <group>'('<group-opts>')'

=> The expansion of contained hosts results to:

   ...
   <host0>'(<host-opts> <glob-opts> <group-opts>')'
   <host1>'(<host-opts> <glob-opts> <group-opts>')'
   ...

=> The expansion of contained nested groups results to:

   ...
   <group-member0>'(<glob-opts>)'('<group-opts>')'
   <group-member1>'(<glob-opts>)'('<group-opts>')'
   ...
```

Figure 15.7: Groups of Stack-Addresses

The context options are applied succesively, thus are 'no-real-context' options, much more a successive superposition. More worst, the GROUP is a set, thus the members of a group are reordered for display and execution purposes frequently. So the context options are - in most practical cases - a required minimum for the attached entity.

15.6 Groups of Stack Addresses

The usage of stacked addresses is supported by the GROUPS objects 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˜**??**.

```
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 15.8: 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˜**??**.

```
  <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 15.9: 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.

.

15.7 ctys-help-on

SYNTAX

<ctys-command>

```
-H <help-option>

   <help-option>:=
         (man|html|pdf)][=((1-9)|<help-on-item>[,<help-on-item-list>])
       | (path|list|listall)
       | funcList=<any-function>][@<module-name>[@...]][,<any-function>...]
       | funcListMod=<any-function>][@<module-name>[@...]][,<any-function>...]
       | funcHead=<any-function>][@<module-name>[@...]][,<any-function>...]
       | _ONLINEHELP_
       | _HELP_
     )
```

DESCRIPTION

The **-H** option is the common generic option of all tools for the display of online help.

The default is the display of man pages within a commandline terminal. This could be any valid document within the search list defined by the variable MANPATH. The output format could be optionally specified as PDF and HTML documents.

This tool is also used within menu entries of the XDG desktop of Freedesktop.org for graphical display of online help. Therefore the current version provides the simple HTML lists **doc.html** for the **DOC-Package**, and the **base.html** file for the **BASE-Package**.

REMARK:
For the **-H** option of the call 'ctys -H man' and 'ctys-vhost -H man' the man parameter is mandatory. In all other cases the call '<any-other-ctys> -H' searches for 'man' output by default within MANPATH.
. **OPTIONS**

The following suboptions and parameters could be applied:

-H path
Displays current document and man path.

-H list
Lists available online documents and manpages.

-H listall
Lists available online documents and manpages including the documents available by MANPATH.

-H (man|html|pdf)[=([1-9]||[<help-on-item-list>])
Displays the requested information with one of the formats **man**, **pdf**, or **html**. The following viewers are preconfigured as shell variables within the configuration files and can be adapted as required:

```
CTYS_MANVIEWER=man
CTYS_PDFVIEWER=(acroread else kpdf else gpdf)
CTYS_HTMLVIEWER=(konqueror else firefox)
```

Default is **manpage** for the current process with **man 1 ...**. Additional constraints could be applied such as another man-section or a filename, which could be either literally matching or a string to be expanded. In case of expansion the first match is taken.

-H funcList[=[<any-function>][@<module-name>[@...]]]
List of function names, sorted by function names. In addition the file names and line numbers are displayed too.

-H funcListMod[=[<any-function>][@<module-name>[@...]]]
List of function names, sorted by file names. In addition the file names and line numbers are displayed too.

-H funcHead[=[<any-function>][@<module-name>[@...]]]
Displays the contents 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.

-H (_ONLINEHELP_|_HELP_)
Displays the predefined online help for the installed package.

. EXAMPLES

<ctys-command> -H (_ONLINEHELP_|_HELP_)
Displays the predefined online help for the installed package.

<ctys-command> -H html=base
Displays a summary of links for all documents contained in the **BASE package**.

<ctys-command> -H html=doc
Displays the extended online help as contained in the **DOC package**.

<ctys-command> -H list
Lists available online documents and manpages.

<ctys-command> -H ctys
Displays the **manpage** for ctys with **man**.

<ctys-command> -H man=ctys
Displays the **manpage** for ctys with **man**.

<ctys-command> -H html=ctys
Displays the **manpage** for ctys with CTYS_HTMLVIEWER, by default **firefox** or **konqueror**.

<ctys-command> -H pdf=ctys
Displays the **manpage** for ctys with CTYS_PDFVIEWER, by default **kpdf**, **gpdf**, or **acroread**.

<ctys-command> -H pdf=howto
Displays the **ctys-howto-online.pdf**, which is displayed in alphabetical order before **ctys-howto-print.pdf**.

<ctys-command> -H pdf=howto-print
Displays the **ctys-howto-print.pdf**, which is the first appropriate match.

<ctys-command> -H pdf=command-ref
Displays the **ctys-command-reference.pdf**.

<ctys-command> -H html=CLI,X11,VNC,VMW

Displays the **manpage** for ctys-CLI, ctys-X11, ctys-VNC and ctys-VNM with
CTYS_HTMLVIEWER, by default **firefox** or **konqueror**. For incomplete names a search with **find** is utilized for name expansion.

<ctys-command> -H html=ctys-extractARPlst,extractMAClst

Displays the **manpage** for ctys-extractARPlst and ctys-extractARPlst with CTYS_HTMLVIEWER, by default **firefox** or **konqueror**. For incomplete names a search with **find** is utilized for name expansion.

Part IV

Appendices

Chapter 16

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!!!

```
------------------------------------------------------------------------------

UnifiedSessionsManager Copyright (C) 2007, 2008, 2010 Arno-Can Uestuensoez

This program comes with ABSOLUTELY NO WARRANTY; for details
refer to provided documentation.
This is free software, and you are welcome to redistribute it
under certain conditions. For details refer to "GNU General Public
License - version 3" <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@users.sourceforge.net
VERSION          = 01_10_011
DATE             = 2010.03.18

LOC              = 117540 CodeLines
LOC-BARE         = 59063 BareCodeLines (no comments and empty lines)
LOD              = 0 DocLines, include LaTeX-sources

TARGET_OS        = Linux:   CentOS/RHEL, Fedora, ScientificLinux,
                            debian, Ubuntu,
                            (gentoo,) mandriva,
                            (knoppix,) (dsl,)
                            SuSE/openSUSE
```

```
                     BSD:     OpenBSD, FreeBSD
                     Solaris: Solaris-10, OpenSolaris
                     Windows: (WNT/Cygwin), (W2K/Cygwin), (WXP/Cygwin),
                              (W2Kx/Cygwin)

TARGET_VM         = KVM, (OpenVZ), QEMU, (VirtualBox,) VMware, Xen
TARGET_WM         = fvwm, Gnome, (KDE,) X11

GUEST_OS          = ANY(some with limited native-acces support)
--------------------------------------------------------------------------
COPYRIGHT         = Arno-Can Uestuensoez - acue@UnifiedSessionsManager.org
LICENCE           = GPL3
--------------------------------------------------------------------------
EXECUTING HOST    = ws2.soho
--------------------------------------------------------------------------
LIBRARIES(static-loaded - generic):

  Nr   Library                                    Version
  -----------------------------------------------------------
  00   bootstrap.01.01.004.sh                     01.10.010
  01   base.sh                                    01.07.001b01
  02   libManager.sh                              01.02.002c01
  03   cli.sh                                     01.07.001b06
  04   misc.sh                                    01.06.001a12
  05   security.sh                                01.06.001a05
  06   help.sh                                    01.10.002
  07   geometry.sh                                01.07.001b06
  08   wmctrlEncapsulation.sh                     01.07.001b06
  09   groups.sh                                  01.11.001
  10   network.sh                                 01.11.001


PLUGINS(dynamic-loaded - ctys specific):

  Nr   Plugin                                     Version
  -----------------------------------------------------------

  00   CORE/CACHE.sh                              01.07.001b01
  01   CORE/CLI.sh                                01.07.001b06
  02   CORE/COMMON.sh                             01.02.002c01
  03   CORE/CONFIG/hook.sh                        01.06.001a14
  04   CORE/DIGGER/hook.sh                        01.07.001b06
  05   CORE/DIGGER/list.sh                        01.02.001b01
  06   CORE/ENV.sh                                01.02.002c01
  07   CORE/EXEC.sh                               01.06.001a15
  08   CORE/GENERIC.sh                            01.10.011
  09   CORE/HELP.sh                               01.02.002c01
  10   CORE/LABELS.sh                             01.07.001b05
  11   CORE/STACKER/hook.sh                       01.07.001b06
  12   CORE/VMs.sh                                01.02.002c01
  13   GENERIC/hook.sh                            01.02.001b01
  14   GENERIC/LIST/list.sh                       01.10.008
  15   GENERIC/ENUMERATE/enumerate.sh             01.02.001b01

  16   HOSTs/CLI/hook.sh                          01.06.001a09
  17   HOSTs/CLI/session.sh                       01.01.001a01
  18   HOSTs/CLI/list.sh                          01.10.008
  19   HOSTs/CLI/info.sh                          01.02.001b01
  20   HOSTs/VNC/hook.sh                          01.02.001b01
  21   HOSTs/VNC/session.sh                       01.06.001a15
  22   HOSTs/VNC/list.sh                          01.07.001b05
  23   HOSTs/VNC/info.sh                          01.02.001b01
  24   HOSTs/X11/hook.sh                          01.06.001a09
  25   HOSTs/X11/session.sh                       01.01.001a01
  26   HOSTs/X11/list.sh                          01.01.001a00
  27   HOSTs/X11/info.sh                          01.02.001b01

  28   VMs/QEMU/hook.sh                           01.10.008
  29   VMs/QEMU/config.sh                         01.01.001a01pre
  30   VMs/QEMU/session.sh                        01.10.008
  31   VMs/QEMU/enumerate.sh                      01.10.008
  32   VMs/QEMU/list.sh                           01.10.008
  33   VMs/QEMU/info.sh                           01.01.001a00pre
  34   VMs/VMW/hook.sh                            01.10.009
  35   VMs/VMW/session.sh                         01.10.009
  36   VMs/VMW/enumerate.sh                       01.06.001a09
  37   VMs/VMW/list.sh                            01.10.009
  38   VMs/VMW/info.sh                            01.02.001b01
  39   VMs/XEN/hook.sh                            01.10.008
  40   VMs/XEN/config.sh                          01.01.001a01
  41   VMs/XEN/session.sh                         01.07.001b06
  42   VMs/XEN/enumerate.sh                       01.01.001a01
  43   VMs/XEN/list.sh                            01.10.008
  44   VMs/XEN/info.sh                            01.01.001a00

  45   PMs/PM/hook.sh                             01.10.008
  46   PMs/PM/session.sh                          01.01.001a00
  47   PMs/PM/enumerate.sh                        01.01.001a01
  48   PMs/PM/list.sh                             01.10.008
  49   PMs/PM/info.sh                             01.01.001a00


CTYS-INTERNAL-SUBCALLS:

  Nr   Component                                  Version
  ------------------------------------------------------------
  00   ctys                                       01_10_011
  01   ctys-callVncserver.sh                      01_10_011
  02   ctys-callVncviewer.sh                      01_10_011
  03   ctys-createConfQEMU.sh                     01_10_011
  04   ctys-distribute.sh                         01_10_011
  05   ctys-dnsutil.sh                          01_10_011
  06   ctys-extractARPlst.sh                    01_10_011
  07   ctys-extractMAClst.sh                    01_10_011
  08   ctys-genmconf.sh                         01_10_011
  09   ctys-groups.sh                           01_10_011
  10   ctys-getMasterPid.sh                     01_10_011
  11   ctys-install.sh                          01_10_009
  12   ctys-install1.sh                         01_10_009
  13   ctys-macros.sh                           01_10_011
  14   ctys-macmap.sh                           01_10_011
  15   ctys-plugins.sh                          01_10_011
  16   ctys-setupVDE.sh                         01_10_011
  17   ctys-smbutil.sh                          01_10_011
  18   ctys-vdbgen.sh                           01_10_011
  19   ctys-vhost.sh                            01_10_011
  20   ctys-vping.sh                            01_10_011
  21   ctys-wakeup.sh                           01_10_011
  22   ctys-xen-network-bridge.sh               01_10_011

  Helpers:

  00   getCPUinfo.sh                            01_10_011
  01   getFSinfo.sh                             01_10_011
  02   getHDDinfo.sh                            01_10_011
  03   getMEMinfo.sh                            01_10_011
  04   getPerfIDX.sh                            01_10_011
  05   getVMinfo.sh                             01_10_011

  Tiny-Helpers:

  00   getCurArch.sh                            OK
  01   getCurCTYSRel.sh                         OK
  02   getCurDistribution.sh                    OK
  03   getCurGID.sh                             OK
  04   getCurOS.sh                              OK
  05   getCurOSRel.sh                           OK
  06   getCurRelease.sh                         OK
  07   getSolarisUUID.sh                        OK
  08   pathlist.sh                              OK


----------------------------------------------------------------------
OPTIONAL/MANDATORY PREREQUISITES:


  bash:GNU bash, version 3.2.25(1)-release (x86_64-redhat-linux-gnu)

  SSH:OpenSSH_4.3p2, OpenSSL 0.9.8e-fips-rhel5 01 Jul 2008

  VNC:VNC Viewer Free Edition 4.1.2 for X - built Mar 24 2009 19:52:30
  wmctrl:wmctrl 1.07



----------------------------------------------------------------------
CURRENT ARG-MEM-USAGE:


  ArgList(bytes):"env|wc -c" => 3284
  ArgList(bytes):"set|wc -c" => 810677
```

Chapter 17

Miscellaneous

17.1 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 Connection-Forward 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.

17.2 PATH

First of all, this is normally just required when handling different versions during development. 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. 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
```

17.3 Configuration files

The configuration of ctys is performed in 4 steps, first has highest priority.

(a) Environment Variable If an environment variable is set, it dominates other settings and it's value is kept.
(b) $HOME/.ctys/ctys.conf Config-File sourced: $HOME/.ctys/ctys.conf
(c) <install dir>/conf/ctys.conf Config-File sourced: <install dir>/conf/ctys.conf
(d) Embedded defaults in ctys.

17.4 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 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 . The standards are available at ieee.org .
The basis for this numbering are the so called DIX and IEEE 802.3 standards. The following items give a short extract:

- 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 The given control bits from the network standards are related to networking, 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

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 18

LICENSES

Additionally a seperate document including all licenses is contained within the package.

ctys-Licenses-01.11-print.pdf

18.1 CCL-3.0 With Attributes

```
License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS
CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS
PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE
WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS
PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND
AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS
LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU
THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH
TERMS AND CONDITIONS.

1. Definitions

   1. "Adaptation" means a work based upon the Work, or upon the Work
   and other pre-existing works, such as a translation, adaptation,
   derivative work, arrangement of music or other alterations of a
   literary or artistic work, or phonogram or performance and includes
   cinematographic adaptations or any other form in which the Work may
   be recast, transformed, or adapted including in any form
   recognizably derived from the original, except that a work that
   constitutes a Collection will not be considered an Adaptation for
   the purpose of this License. For the avoidance of doubt, where the
   Work is a musical work, performance or phonogram, the
   synchronization of the Work in timed-relation with a moving image
   ("synching") will be considered an Adaptation for the purpose of
   this License.

   2. "Collection" means a collection of literary or artistic works,
```

such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined above) for the purposes of this License.

3. "Distribute" means to make available to the public the original and copies of the Work through sale or other transfer of ownership.

4. "Licensor" means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.

5. "Original Author" means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.

6. "Work" means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.

7. "You" means an individual or entity exercising rights under this License who has not previously violated the terms of this License

with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

8. "Publicly Perform" means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.

9. "Reproduce" means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work, including storage of a protected performance or phonogram in digital form or other electronic medium.

2. Fair Dealing Rights. Nothing in this License is intended to reduce, limit, or restrict any uses free from copyright or rights arising from limitations or exceptions that are provided for in connection with the copyright protection under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

1. to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections; and,

2. to Distribute and Publicly Perform the Work including as incorporated in Collections.

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats, but otherwise you have no rights to make Adaptations. Subject to 8(f), all rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights set forth in Section 4(d).

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

1. You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform

sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.

4.

For the avoidance of doubt:

1. Non-waivable Compulsory License Schemes. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License;

2. Waivable Compulsory License Schemes. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License if Your exercise of such rights is for a purpose or use which is otherwise than noncommercial as permitted under Section 4(b) and otherwise waives the right to collect royalties through any statutory or compulsory licensing scheme; and,

3. Voluntary License Schemes. The Licensor reserves the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License that is for a purpose or use which is otherwise than noncommercial as permitted under Section 4(b).

5. Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO

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

FreeDOS

[128] *FreeDOS*

[129] *Balder*

Dynagen/Dynamips

[130] Dynagen, by Greg Anuzelli:
Dynagen

[131] Dynamips, Cisco(TM) 7200 Simulator:
Dynamips, Cisco(TM) 7200 Simulator

QEMU-Networking with VDE

[132] VDE-Virtual Distributed Ethernet:
http://sourceforge.net/projects/vde

[133] VirtualSquare: *http://virtualsquare.org*

[134] VirtualSquare: *Basic Networking*

PXE

[135] By H. Peter Anvin: *SYSLINUX - PXELINUX -ISOLINUX*

[136] PXE-ROM-Images: *Etherboot*

Routing

[137] "Linux Advanced Routing&Traffic Control", by Bert Hubert:
LARTC-HOWTO

Scratchbox

[138] *Scratchbox*

Serial-Console

[139] By van Emery: *"Linux Serial Console HOWTO"*

[140] By David S. Lawyer / Greg Hawkins: *"Serial-HOWTO"*

[141] By David S. Lawyer: *"Text-Terminal-HOWTO"*

Miscellaneous

[142] IEEE: *http://www.ieee.org*

[143] The GNU Netcat: *Netcat*

[144] Netcat Wikipedia: *Netcat Wikipedia*

[145] By van Emery: *"Linux Gouge"*

UnfiedSessionsManager Versions

[146] The first public version of 2008.02.11, by Arno-Can Uestuensoez. Available as online help only by "ctys -H print"(more than 230pg. as ACII-Only): *"UnifiedSesionsManager" http://sourceforge.net/projects/ctys*

[147] The second public version of 2008.07.10, by Arno-Can Uestuensoez: *"UnifiedSesionsManager" http://sourceforge.net/projects/ctys*

[148] The second public version with minor updates of 2008.08.6, by Arno-Can Uestuensoez: *"UnifiedSesionsManager" http://sourceforge.net/projects/ctys*

[149] Minor editorial updates of 2008.08.12, by Arno-Can Uestuensoez: *"UnifiedSesionsManager" http://sourceforge.net/projects/ctys*

[150] Enhanced documentation, 2008.08.16, by Arno-Can Uestuensoez: *"UnifiedSesionsManager" http://sourceforge.net/projects/ctys*

[151] Major enhancements and feature updates, beginning 2010/02, by Arno-Can Uestuensoez: *"UnifiedSesionsManager" http://sourceforge.net/projects/ctys*

Sponsored OpenSource Projects

Support is available exclusively by direct contact only.

[152] Ingenieurbuero Arno-Can Uestuensoez - OpenSource:
http://www.i4p.org

[153] UnifiedSessionsManager:
http://www.UnifiedSessionsManager.org
http://sourceforge.net/projects/ctys
http://ctys.berlios.de

Commercial Support

Commercial support and additional services are available exclusively by direct contact only.

[154] Ingenieurbuero Arno-Can Uestuensoez:
http://www.i4p.com
http://www.i4p.de
http://www.i4p.eu