

WIND RIVER

Wind River® Simics®

REFERENCE MANUAL

4.6

Revision
Date

4081
2012-11-16

Copyright © 2010–2012 Wind River Systems, Inc.

All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means without the prior written permission of Wind River Systems, Inc.

Wind River, Simics, Tornado, and VxWorks are registered trademarks of Wind River Systems, Inc. The Wind River logo is a trademark of Wind River Systems, Inc. Any third-party trademarks referenced are the property of their respective owners. For further information regarding Wind River trademarks, please see:
www.windriver.com/company/terms/trademark.html

This product may include software licensed to Wind River by third parties. Relevant notices (if any) are provided in your product installation at the following location:
installDir/LICENSES-THIRD-PARTY/.

Wind River may refer to third-party documentation by listing publications or providing links to third-party Web sites for informational purposes. Wind River accepts no responsibility for the information provided in such third-party documentation.

Corporate Headquarters

Wind River
500 Wind River Way
Alameda, CA 94501-1153
U.S.A.

Toll free (U.S.A.): 800-545-WIND
Telephone: 510-748-4100
Facsimile: 510-749-2010

For additional contact information, see the Wind River Web site:

www.windriver.com

For information on how to contact Customer Support, see:

www.windriver.com/support

Contents

1	Introduction	35
2	Command-Line Tools and Programs	36
2.1	simics	36
2.2	addon-manager	40
2.3	craff	41
2.4	install-simics	43
2.5	workspace-setup	44
3	Commands	46
3.1	Complete List	46
3.2	By Categories	85
	Breakpoints	85
	CD-ROM	85
	Calypso	86
	Changing Simulated State	86
	Command Line Interface	86
	Components	89
	Configuration	94
	Debugging	95
	Disk	97
	Ethernet	97
	Execution	98
	Fibre Channel	100
	Files and Directories	100
	GUI	100
	Generic Link	100
	Graphics Adapter	101
	Graphics Console	101
	Haps	101
	Help	101
	IDE	101
	Inspecting Simulated State	102
	Keypad	102
	Logging	102

MMC	102
Memory	103
Modules	104
Networking	104
Output	105
PCI	106
PCMCIA	106
Profiling	106
Python	107
RapidIO Link	107
Real Network	107
Registers	108
Reverse Execution	108
SATA	109
SCSI	109
Serial Link	109
Simics Search Path	109
Speed	110
Status Panel	110
Symbolic Debugging	110
Test	112
Text Console	112
Tracing	112
USB	112
VMP	113
3.3 Global Command Descriptions	113
4 Components	312
cell-and-clocks	313
Attributes	313
Class Attributes	314
Command List	314
Command Descriptions	314
component	316
Attributes	316
Class Attributes	316
Command List	317
Command Descriptions	317
cp3_quad100tx	320
Attributes	320
Command List	320
Command Descriptions	320
datagram_link	322
Attributes	322
Command List	322
Command Descriptions	322

ddr-memory-module	324
Attributes	324
Class Attributes	325
Command List	325
Command Descriptions	325
ddr2-memory-module	327
Attributes	327
Class Attributes	328
Command List	328
Command Descriptions	328
ddr3-memory-module	330
Attributes	330
Class Attributes	331
Command List	331
Command Descriptions	331
dummy-component	333
Attributes	333
Class Attributes	334
Command List	334
Command Descriptions	334
etg_comp	335
Attributes	335
Command List	335
Command Descriptions	335
etg_panel	337
Command List	337
Command Descriptions	337
eth_injector_comp	338
Command List	338
Command Descriptions	338
ethernet_cable	339
Attributes	339
Command List	339
Command Descriptions	340
ethernet_hub	345
Attributes	345
Command List	345
Command Descriptions	346
ethernet_switch	351
Attributes	351
Command List	351
Command Descriptions	352
ethernet_vlan_switch	357
Attributes	357
Command List	357

Command Descriptions	358
example-keypad	362
Attributes	362
Class Attributes	362
Command List	363
Command Descriptions	363
example-status-panel	364
Attributes	364
Class Attributes	364
Command List	364
Command Descriptions	365
generic_PCIE_switch	366
Command List	366
Command Descriptions	366
i2c_link_v2	367
Attributes	367
Command List	367
Command Descriptions	367
instruction-data-splitter	368
Attributes	368
Class Attributes	368
Command List	369
Command Descriptions	369
isa-fourport	370
Attributes	370
Class Attributes	370
Command List	371
Command Descriptions	371
isa-lance	372
Attributes	372
Class Attributes	373
Command List	373
Command Descriptions	373
isa-vga	374
Attributes	374
Class Attributes	374
Command List	375
Command Descriptions	375
memory-timer	376
Attributes	376
Class Attributes	376
Command List	377
Command Descriptions	377
micron_mtfc2ggqdi_emmc_card	378
Command List	378

Command Descriptions	378
micron_mtfc4ggqdi_emmc_card	379
Command List	379
Command Descriptions	379
micron_mtfc4ggqdi_sdhc_card	380
Command List	380
Command Descriptions	380
os_awareness	381
Command List	381
Command Descriptions	381
pc-dual-serial-ports	392
Attributes	392
Class Attributes	392
Command List	393
Command Descriptions	393
pc-floppy-controller	394
Attributes	394
Class Attributes	394
Command List	395
Command Descriptions	395
pc-quad-serial-ports	396
Attributes	396
Class Attributes	396
Command List	397
Command Descriptions	397
pc-single-parallel-port	398
Attributes	398
Class Attributes	398
Command List	399
Command Descriptions	399
pci-accel-vga	400
Attributes	400
Class Attributes	400
Command List	401
Command Descriptions	401
pci-am79c973	402
Attributes	402
Class Attributes	402
Command List	403
Command Descriptions	403
pci-bcm5703c	404
Attributes	404
Class Attributes	404
Command List	405
Command Descriptions	405

pci-bcm5704c	406
Attributes	406
Class Attributes	407
Command List	407
Command Descriptions	407
pci-dec21041	408
Attributes	408
Class Attributes	408
Command List	409
Command Descriptions	409
pci-dec21140a	410
Attributes	410
Class Attributes	410
Command List	411
Command Descriptions	411
pci-dec21140a-dml	412
Attributes	412
Class Attributes	412
Command List	413
Command Descriptions	413
pci-dec21143	414
Attributes	414
Class Attributes	414
Command List	415
Command Descriptions	415
pci-i21152	416
Attributes	416
Class Attributes	416
Command List	417
Command Descriptions	417
pci-i82543gc	418
Attributes	418
Class Attributes	418
Command List	419
Command Descriptions	419
pci-i82546bg	420
Attributes	420
Class Attributes	420
Command List	421
Command Descriptions	421
pci-i82559	422
Attributes	422
Class Attributes	422
Command List	423
Command Descriptions	423

pci-isp1040	424
Attributes	424
Class Attributes	424
Command List	425
Command Descriptions	425
pci-isp2200	426
Attributes	426
Class Attributes	426
Command List	427
Command Descriptions	427
pci-pd6729	428
Attributes	428
Class Attributes	428
Command List	429
Command Descriptions	429
pci-pmc1553	430
Attributes	430
Class Attributes	430
Command List	431
Command Descriptions	431
pci-sil680a	432
Attributes	432
Class Attributes	432
Command List	433
Command Descriptions	433
pci-sym53c810	434
Attributes	434
Class Attributes	434
Command List	435
Command Descriptions	435
pci-sym53c875	436
Attributes	436
Class Attributes	436
Command List	437
Command Descriptions	437
pci-sym53c876	438
Attributes	438
Class Attributes	438
Command List	439
Command Descriptions	439
pci-tsb12lv26	440
Attributes	440
Class Attributes	440
Command List	441
Command Descriptions	441

pci-vga	442
Attributes	442
Class Attributes	442
Command List	443
Command Descriptions	443
phy-mii-transceiver	444
Attributes	444
Class Attributes	444
Command List	445
Command Descriptions	445
phy_comp	446
Command List	446
Command Descriptions	446
ps2-keyboard-mouse	447
Attributes	447
Class Attributes	447
Command List	448
Command Descriptions	448
rapidio_link	449
Attributes	449
Command List	449
Command Descriptions	449
real-network-bridge	450
Attributes	450
Class Attributes	450
Command List	451
Command Descriptions	451
real-network-host	452
Attributes	452
Class Attributes	452
Command List	453
Command Descriptions	453
real-network-router	454
Attributes	454
Class Attributes	454
Command List	455
Command Descriptions	455
sample-gcache	456
Attributes	456
Class Attributes	456
Command List	457
Command Descriptions	457
sDRAM-memory-module	458
Attributes	458
Class Attributes	459

Command List	459
Command Descriptions	459
ser_link	461
Attributes	461
Command List	461
Command Descriptions	461
signal_link	462
Attributes	462
Command List	462
Command Descriptions	462
simple-fc-disk	464
Attributes	464
Class Attributes	465
Command List	465
Command Descriptions	465
simple_memory_module	466
Command List	466
Command Descriptions	466
sio-w83627hf	467
Attributes	467
Class Attributes	467
Command List	468
Command Descriptions	468
std-etg	469
Attributes	469
Class Attributes	469
Command List	470
Command Descriptions	470
std-ethernet-link	471
Attributes	471
Class Attributes	472
Command List	472
Command Descriptions	472
std-firewire-bus	473
Attributes	473
Class Attributes	473
Command List	474
Command Descriptions	474
std-firewire-sample-device	475
Attributes	475
Class Attributes	475
Command List	476
Command Descriptions	476
std-generic-link	477
Attributes	477

Class Attributes	478
Command List	478
Command Descriptions	478
std-generic-link-sample	479
Attributes	479
Class Attributes	479
Command List	480
Command Descriptions	480
std-graphics-console	481
Attributes	481
Class Attributes	482
Command List	482
Command Descriptions	482
std-host-serial-console	484
Attributes	484
Class Attributes	484
Command List	485
Command Descriptions	485
std-ide-cdrom	486
Attributes	486
Class Attributes	486
Command List	487
Command Descriptions	487
std-ide-disk	488
Attributes	488
Class Attributes	488
Command List	489
Command Descriptions	489
std-ms1553-link	490
Attributes	490
Class Attributes	490
Command List	491
Command Descriptions	491
std-pcmcia-flash-disk	492
Attributes	492
Class Attributes	492
Command List	493
Command Descriptions	493
std-scsi-bus	494
Attributes	494
Class Attributes	494
Command List	495
Command Descriptions	495
std-scsi-cdrom	496
Attributes	496

Class Attributes	496
Command List	497
Command Descriptions	497
std-scsi-disk	498
Attributes	498
Class Attributes	498
Command List	499
Command Descriptions	499
std-serial-link	500
Attributes	500
Class Attributes	501
Command List	501
Command Descriptions	501
std-server-console	502
Attributes	502
Class Attributes	502
Command List	503
Command Descriptions	503
std-service-node	504
Attributes	504
Class Attributes	504
Command List	504
Command Descriptions	505
std-super-io	510
Attributes	510
Class Attributes	510
Command List	511
Command Descriptions	511
std-telnet-console	512
Attributes	512
Class Attributes	512
Command List	513
Command Descriptions	513
std-text-console	514
Attributes	514
Class Attributes	515
Command List	515
Command Descriptions	515
std-text-graphics-console	517
Attributes	517
Class Attributes	518
Command List	518
Command Descriptions	518
std_mmc_card	520
Attributes	520

Command List	520
Command Descriptions	520
std_sata_cdrom	521
Command List	521
Command Descriptions	521
std_sata_disk	522
Attributes	522
Command List	522
Command Descriptions	522
top-component	524
Attributes	524
Class Attributes	525
Command List	525
Command Descriptions	525
usb-disk	526
Attributes	526
Class Attributes	526
Command List	527
Command Descriptions	527
usb-santa	528
Attributes	528
Class Attributes	528
Command List	529
Command Descriptions	529
usb-tablet-comp	530
Attributes	530
Class Attributes	530
Command List	531
Command Descriptions	531
usb-wacom-tablet-comp	532
Attributes	532
Class Attributes	532
Command List	533
Command Descriptions	533
usb_hs_keyboard_comp	534
Command List	534
Command Descriptions	534
usb_keyboard_comp	535
Command List	535
Command Descriptions	535
usb_mouse_comp	536
Command List	536
Command Descriptions	536
usb_xmas_tree	537
Command List	537

Command Descriptions	537
5 Classes	538
accel-vga	539
Attributes	539
Command List	539
Command Descriptions	539
AM79C973	542
Attributes	542
Command List	543
Command Descriptions	543
AT24Cxx	545
Attributes	545
Command List	545
Command Descriptions	546
attr-meter	547
Attributes	547
Command List	547
Command Descriptions	547
base-trace-mem-hier	548
Command List	548
Command Descriptions	548
BCM5703C	550
Attributes	550
Command List	551
Command Descriptions	551
BCM5704C	553
Attributes	553
Command List	554
Command Descriptions	554
bitmask-translator	556
Attributes	556
Command List	556
Command Descriptions	556
branch_recorder	557
Attributes	557
Command List	557
breakpoints	558
cell	559
Attributes	559
Command List	559
Command Descriptions	559
central-client	561
Command List	561
Command Descriptions	561
central-server	563

Command List	563
Command Descriptions	563
CL-PD6729	564
Command List	564
Command Descriptions	564
cli	566
clipboard-gateway	567
Attributes	567
Command List	567
Command Descriptions	567
clock	568
Attributes	568
Command List	568
Command Descriptions	568
connector	569
Attributes	569
Command List	569
Command Descriptions	570
context	571
Command List	571
Command Descriptions	571
coverage_profiler	578
Attributes	578
Command List	578
Command Descriptions	578
cpu-group	580
Attributes	580
Command List	580
Command Descriptions	580
data-profiler	581
Command List	581
Command Descriptions	581
datagram_link_endpoint	582
Attributes	582
Command List	582
Command Descriptions	582
datagram_link_impl	583
Attributes	583
Command List	583
Command Descriptions	583
DEC21041	584
Attributes	584
Command List	585
Command Descriptions	585
DEC21140A	587

Attributes	587
Command List	588
Command Descriptions	588
DEC21140A-dml	590
Attributes	590
Command List	591
Command Descriptions	591
DEC21143	592
Attributes	592
Command List	593
Command Descriptions	593
disassemble_v9	595
disassemble_x86	596
dm9161	597
Attributes	597
Command List	598
Command Descriptions	598
dynamic_link_connector	599
Attributes	599
Command List	600
Command Descriptions	600
empty-device-c	601
Command List	601
Command Descriptions	601
empty-device-cc	602
Command List	602
Command Descriptions	602
etg	603
Attributes	603
Command List	603
Command Descriptions	603
eth-cable-link	605
Attributes	605
Command List	605
Command Descriptions	605
eth-cable-link-endpoint	606
Attributes	606
Command List	606
Command Descriptions	606
eth-hub-link	607
Attributes	607
Command List	607
Command Descriptions	607
eth-hub-link-endpoint	608
Attributes	608

Command List	608
Command Descriptions	608
eth-link-snoop-endpoint	609
Attributes	609
Command List	609
Command Descriptions	609
eth-probe	610
Attributes	610
Command List	610
Command Descriptions	610
eth-probe-port	613
Attributes	613
eth-switch-link	614
Attributes	614
Command List	614
Command Descriptions	614
eth-switch-link-endpoint	615
Attributes	615
Command List	615
Command Descriptions	615
eth-switch-link-snoop-endpoint	617
Attributes	617
Command List	617
Command Descriptions	617
eth-transceiver	619
Attributes	619
Command List	619
Command Descriptions	619
eth_injector	621
Command List	621
Command Descriptions	621
ethernet-link	622
Attributes	622
Command List	624
Command Descriptions	624
fake-space	628
Attributes	628
Command List	628
fc-disk	629
Attributes	629
Command List	629
Command Descriptions	630
file-cdrom	633
Attributes	633
Command List	633

Command Descriptions	633
firewire_bus	634
Attributes	634
Command List	634
Command Descriptions	634
firewire_device_test_wrapper	636
Attributes	636
firewire_sample_device	637
Command List	637
Command Descriptions	637
flash-memory	638
Attributes	638
frequency_bus	639
Attributes	639
Command List	639
Command Descriptions	640
frontend-server	641
frontend-server-console	642
ftp-alg	643
Attributes	643
ftp-control	644
Attributes	644
Command List	644
Command Descriptions	644
ftp-data	645
Attributes	645
Command List	645
Command Descriptions	645
ftp-service	646
Attributes	646
Command List	647
Command Descriptions	647
g-cache	648
Attributes	648
Command List	648
Command Descriptions	648
gdb-remote	651
Command List	652
Command Descriptions	653
generic-flash-memory	654
Attributes	654
Command List	656
Command Descriptions	656
generic-message-link	658
Attributes	658

generic-message-sample-device	660
Attributes	660
generic mmc-card	661
Command List	661
Command Descriptions	661
generic eth phy	662
Attributes	662
Command List	663
Command Descriptions	663
generic pcie switch port	664
Attributes	664
Command List	665
Command Descriptions	665
generic spi flash	666
Attributes	666
Command List	667
Command Descriptions	667
gfx-console	668
Attributes	668
Command List	671
Command Descriptions	671
gui	676
hap-meter	677
Attributes	677
Command List	677
Command Descriptions	677
host-condor-pci-1553	679
host-pmc-1553	680
host-serial-console	681
Attributes	681
Command List	681
Command Descriptions	681
hostfs	683
Command List	683
Command Descriptions	683
hypersim-pattern-matcher	684
Attributes	684
Command List	684
Command Descriptions	684
i21150	686
Attributes	686
Command List	686
Command Descriptions	686
i21152	688
Attributes	688

Command List	688
Command Descriptions	688
i21154	690
Attributes	690
Command List	690
Command Descriptions	691
i21554-prim	692
Attributes	692
i21554-scnd	693
Attributes	693
i21555-prim	694
Attributes	694
i21555-scnd	695
Attributes	695
i2c-bus	696
Attributes	696
Command List	697
Command Descriptions	697
i2c-link-endpoint	698
Attributes	698
Command List	698
Command Descriptions	698
i2c-link-impl	699
Attributes	699
Command List	699
Command Descriptions	699
i2c_link_v1	700
Attributes	700
Command List	700
Command Descriptions	701
i2c_slave_v2_to_bus_adapter	702
Attributes	702
Command List	702
Command Descriptions	702
i2c_wire	703
Attributes	703
Command List	703
Command Descriptions	703
i82543	704
Attributes	704
Command List	705
Command Descriptions	705
i82546	707
Attributes	707
Command List	708

	Command Descriptions	708
i82559	710
	Attributes	710
	Command List	711
	Command Descriptions	711
id-splitter	712
ide	713
	Attributes	713
	Command List	714
	Command Descriptions	714
ide-cdrom	715
	Attributes	715
	Command List	716
	Command Descriptions	716
ide-disk	718
	Attributes	718
	Command List	719
	Command Descriptions	719
image	724
	Attributes	724
	Command List	725
	Command Descriptions	725
ISP1040	728
	Attributes	728
	Command List	728
	Command Descriptions	728
ISP2200	730
	Attributes	730
	Command List	730
	Command Descriptions	730
mac_splitter	732
	Attributes	732
mem-traffic-meter	733
	Command List	733
	Command Descriptions	733
memory-space	734
	Attributes	734
	Command List	735
	Command Descriptions	736
microwire-eeprom	742
mii-management-bus	743
	Attributes	743
	Command List	743
	Command Descriptions	743
mii-transceiver	744

Attributes	744
Command List	745
Command Descriptions	745
ms1553-link	747
Command List	747
Command Descriptions	747
ms1553-recorder-bm	749
ms1553-test-rt	750
Attributes	750
ms1553_rt	751
Attributes	751
mtprof	752
Command List	752
Command Descriptions	752
NS16450	755
Attributes	755
Command List	755
Command Descriptions	755
NS16550	757
Attributes	757
Command List	757
Command Descriptions	757
NS16550_c	759
Command List	759
Command Descriptions	759
onfi_flash	760
Attributes	761
Command List	762
Command Descriptions	763
PCF8582C	764
Attributes	764
pci-bus	765
Attributes	765
Command List	766
Command Descriptions	766
pcie-bus	767
Attributes	767
Command List	768
Command Descriptions	768
perfanalyze-client	769
persistent-ram	770
Attributes	770
Command List	770
Command Descriptions	770
PMC1553	771

Attributes	771
Command List	772
Command Descriptions	773
port-forward-incoming-server	774
Attributes	774
port-forward-outgoing-server	775
Attributes	775
Command List	775
Command Descriptions	775
port-space	777
Attributes	777
Command List	777
Command Descriptions	778
preferences	782
Attributes	782
Command List	784
Command Descriptions	785
ram	786
Attributes	786
Command List	786
Command Descriptions	786
rapdio_link_endpoint	787
Attributes	787
Command List	787
Command Descriptions	787
rapdio_link_impl	789
Attributes	789
Command List	789
Command Descriptions	789
rapdio_simple_device	790
Attributes	790
rapdio_tape	791
Command List	791
Command Descriptions	791
realtime	793
Attributes	793
Command List	794
Command Descriptions	794
recorder	796
Command List	796
Command Descriptions	796
remote_sync_domain	798
Attributes	798
Command List	798
Command Descriptions	798

remote_sync_node	799
Command List	799
Command Descriptions	799
remote_sync_server	800
Attributes	800
Command List	800
Command Descriptions	800
rev-execution	801
rn-eth-bridge-raw	802
Attributes	802
Command List	802
Command Descriptions	802
rn-eth-bridge-tap	803
Attributes	803
Command List	803
Command Descriptions	803
rn-eth-proxy-raw	805
Attributes	805
Command List	805
Command Descriptions	805
rn-ip-router-raw	806
Attributes	806
Command List	806
Command Descriptions	806
rom	807
Attributes	807
Command List	807
Command Descriptions	807
scsi-bus	808
Attributes	808
Command List	808
Command Descriptions	808
scsi-cdrom	809
Attributes	809
Command List	810
Command Descriptions	810
scsi-disk	812
Attributes	812
Command List	813
Command Descriptions	813
selfprof	818
Command List	818
Command Descriptions	818
ser-link-endpoint	819
Attributes	819

Command List	819
Command Descriptions	819
ser-link-impl	820
Attributes	820
Command List	820
Command Descriptions	820
serial-link	821
Attributes	821
Command List	822
Command Descriptions	822
service-node	823
Attributes	823
Command List	826
Command Descriptions	827
service-node-device	831
Attributes	831
Command List	832
Command Descriptions	832
set-memory	833
Attributes	833
Command List	833
Command Descriptions	833
signal-bus	834
Attributes	834
Command List	834
Command Descriptions	834
signal_link_endpoint	835
Attributes	835
Command List	835
Command Descriptions	835
signal_link_impl	837
Attributes	837
Command List	837
Command Descriptions	837
signal_to_interrupt	838
Attributes	838
Command List	838
Command Descriptions	838
SIL680A	839
Command List	839
Command Descriptions	839
sim	841
Attributes	841
Command List	843
Command Descriptions	844

simple-scsi-disk	845
Attributes	845
Command List	845
Command Descriptions	845
slaver	850
Attributes	850
Command List	850
Command Descriptions	850
software_tracker	851
Attributes	851
Command List	851
Command Descriptions	851
state-assertion	852
Command List	852
Command Descriptions	852
static_link_connector	855
Attributes	855
Command List	856
Command Descriptions	856
status_panel	857
Attributes	857
status_panel_view	858
Attributes	858
SYM53C810	859
Attributes	859
Command List	859
Command Descriptions	859
SYM53C875	861
Attributes	861
Command List	861
Command Descriptions	861
symtable	863
Command List	863
Command Descriptions	863
sync_domain	867
Command List	867
Command Descriptions	867
system_panel_bool_in	868
Attributes	868
Command List	868
Command Descriptions	868
system_panel_bool_out	869
Attributes	869
Command List	869
Command Descriptions	869

system_panel_number_in	870
Attributes	870
Command List	870
Command Descriptions	870
system_panel_number_out	871
Attributes	871
Command List	871
Command Descriptions	871
system_panel_state_manager	872
Attributes	872
Command List	872
Command Descriptions	872
tcf-agent	873
Attributes	873
Command List	873
Command Descriptions	873
tcf-context-proxy	874
Attributes	874
Command List	874
Command Descriptions	874
telnet_console	876
Attributes	876
Command List	876
Command Descriptions	877
telnet_frontend	882
Attributes	882
Command List	882
Command Descriptions	882
terminal_frontend	884
Attributes	884
text-console	885
Attributes	885
Command List	886
Command Descriptions	887
text_panel_frontend	894
Command List	894
Command Descriptions	894
time-server	895
Attributes	895
trace-mem-hier	896
trace-sync	897
Attributes	897
Command List	897
Command Descriptions	897
trans-sorter	899

	Attributes	899
trans-splitter		900
	Attributes	900
trans-staller		901
TSB12LV26		902
	Attributes	902
	Command List	902
	Command Descriptions	903
tsi500		904
	Attributes	904
	Command List	904
	Command Descriptions	904
usb_disk		905
	Attributes	905
usb_hs_keyboard		907
	Attributes	907
	Command List	908
	Command Descriptions	908
usb_keyboard		910
	Attributes	910
	Command List	911
	Command Descriptions	911
usb_mouse		913
	Attributes	913
	Command List	913
	Command Descriptions	914
usb_santa		915
	Attributes	915
usb_tablet		916
	Attributes	916
	Command List	916
	Command Descriptions	917
usb_wacom_tablet		918
	Attributes	918
	Command List	918
	Command Descriptions	919
vga_pci		920
	Attributes	920
	Command List	920
	Command Descriptions	921
vmcom		923
	Attributes	923
	Command List	923
	Command Descriptions	923
wdb-remote		925

Attributes	925
Command List	926
Command Descriptions	926
6 Interfaces	927
abs_pointer	928
abs_pointer_activate	929
address_profiler	930
Command Descriptions	931
attribute_monitor	933
branch_arc	934
breakpoint	936
Command Descriptions	937
bridge	939
checkpoint	940
component	941
component_connector	944
connector	946
cycle	948
Command Descriptions	951
datagram_link	953
ethernet_cable	954
ethernet_common	955
ethernet_device	957
ethernet_link	960
ethernet_probe	963
ethernet_snoop	965
ethernet_vlan_snoop	966
extended_serial	967
firewire_bus	968
firewire_device	970
frequency_listener	972
generic_message_device	973
generic_message_link	974
i2c_bridge	977
i2c_bus	978
i2c_device	980
i2c_link	981
i2c_master	986
i2c_master_v2	987
i2c_slave	988
i2c_slave_v2	990
ieee_802_3_mac	993
ieee_802_3_mac_v3	994
ieee_802_3_phy	995
ieee_802_3_phy_v2	996

ieee_802_3_phy_v3	997
image	998
int_register	1000
Command Descriptions	1001
io_memory	1006
keyboard	1008
log_object	1009
map_demap	1011
mdio45_bus	1013
mdio45_phy	1014
memory_space	1015
Command Descriptions	1016
microwire	1017
mii	1018
mii_management	1019
mmc	1020
ms1553_bridge_terminal	1021
ms1553_link	1022
ms1553_terminal	1024
nand_flash	1025
path_translate	1026
pci_express	1027
pci_express_hotplug	1029
port_space	1030
rapidio_v3	1031
rapidio_v5	1033
recorder	1036
recorder_v2	1037
remote_callback	1038
rs232_device	1039
scale_factor_listener	1040
serial_device	1041
serial_link	1042
serial_peripheral_interface_slave	1043
signal	1044
simple_dispatcher	1046
simple_interrupt	1047
slaver_message	1048
snoop_memory	1049
software	1050
symtable	1054
timing_model	1056
translate	1057
uint64_state	1058
x86_cpuid	1059

7 Haps	1060
Central_Blocked	1060
Central_Unblocked	1060
CLI_Command_Added	1060
CLI_Variable_Write	1061
Component_Change	1061
Component_Hierarchy_Change	1061
Core_Address_Not_Mapped	1062
Core_Asynchronous_Trap	1062
Core_At_Exit	1062
Core_Back_To_Front	1063
Core_Breakpoint_Change	1063
Core_Breakpoint_Memop	1063
Core_Conf_Class_Register	1064
Core_Conf_Class_Unregister	1064
Core_Conf_Clock_Change_Cell	1064
Core_Conf_Object_Change_Clock	1064
Core_Conf_Object_Create	1065
Core_Conf_Object_Created	1065
Core_Conf_Object_Delete	1065
Core_Conf_Object_Pre_Delete	1066
Core_Conf_Object_Rename	1066
Core_Conf_Objects_Created	1066
Core_Conf_Objects_Deleted	1066
Core_Configuration_Loaded	1067
Core_Context_Activate	1067
Core_Context_Change	1067
Core_Context_Deactivate	1068
Core_Context_Updated	1068
Core_Continuation	1068
Core_Control_Register_Read	1069
Core_Control_Register_Write	1069
Core_Cycle_Count	1070
Core_Device_Access_Memop	1070
Core_Disable_Breakpoints	1070
Core_Discard_Future	1070
Core_DSTC_Flush_Counter	1071
Core_Exception	1071
Core_Exception_Return	1072
Core_External_Interrupt	1072
Core_Frequency_Changed	1073
Core_Global_Message	1073
Core_Hap_Callback_Installed	1073
Core_Hap_Callback_Removed	1073
Core_Hap_Type_Added	1074

Core_Image_Activity	1074
Core_Initial_Configuration	1074
Core_Log_Message	1075
Core_Log_Message_Extended	1075
Core_Magic_Instruction	1075
Core_Memory_Space_Map_Changed	1076
Core_Mode_Change	1076
Core_Module_Loaded	1076
Core_Multithreading_Changed	1077
Core_NotImplemented	1077
Core_Periodic_Event	1077
Core_Preferences_Changed	1077
Core_Processor_Schedule_Changed	1078
Core_Recent_Files_Changed	1078
Core_Rexec_Active	1078
Core_Simulation_Mode_Change	1079
Core_Simulation_Stopped	1079
Core_SkipTo_Progress	1079
Core_Source_Step	1080
Core_Step_Count	1080
Core_Sync_Instruction	1080
Core_Time_Transition	1081
Core_Timing_Model_Change	1081
Core_Workspace_Changed	1081
Core_Write_Configuration	1082
Ethernet_Frame	1082
FC_SCSI_Command	1082
Firewire_Reset	1083
Firewire_Transfer	1083
Generic_Message_Frame	1083
Gfx_Break_String	1084
Graphics_Console_New_Title	1084
Graphics_Console_Show_Hide	1084
Internal_Bookmark_List_Changed	1084
Internal_Micro_Checkpoint_Loaded	1085
Internal_SB_Barrier	1085
Internal_SB_Command_File	1085
Internal_SB_Cycle_Count	1086
Internal_SB_Log	1086
Internal_SB_Pipe	1086
Internal_SB_Step_Count	1087
Internal_SB_Time	1087
Internal_SB_Variable	1087
Internal_Time_Direction_Changed	1088
Internal_Time_Quantum_Changed	1088

Rexec_Limit_Exceeded	1088
SCSI_CDROM_Command	1088
SCSI_Disk_Command	1089
Symtable_Updated	1089
Text_Console_New_Title	1089
Text_Console_Show_Hide	1090
UI_Run_State_Changed	1090
Vga_Break_String	1090
Xterm_Break_String	1091
8 Modules	1092
Index	1144

Chapter 1

Introduction

This manual contains the reference documentation for Simics itself. It includes a complete list of all commands defined by the Simics core as well as by the loadable modules included in the main product. It also provides documentation on the available components and classes. Finally, it describes the functions, interfaces, and haps that are available for scripting.

This manual is aimed at normal Simics users (the command list chapter, and possibly the lists of components and classes) and script developers (haps, classes, and interfaces).

This manual is strictly meant to be used as reference documentation, and it provides little explanation on how to use the various features described. Refer to other Simics manuals for more information; these are listed in the *Documentation Contents* document.

Additional reference documentation:

Model Builder Reference Manual

Includes the *Device API* and other information for developers writing models of hardware. This manual is part of the *Wind River Simics Model Builder* product.

Extension Builder Reference Manual

Includes the *Simulator API* intended for developers writing simulator extensions. This manual is part of the *Wind River Simics Extension Builder* product.

Chapter 2

Command-Line Tools and Programs

This section documents the command-line options of the tools and programs delivered as part of Simics.

2.1 simics

Options

-h

Makes Simics print a brief help screen and exit.

-obscure

Prints out a summary of advanced command line flags and exit.

-no-win

Disable external windows. This will prevent Simics from opening target console windows or any other external windows.

Disabling a target console only makes it invisible. It does not change its simulated functionality. Disabling it is useful when running in batch mode or when the target console isn't needed, since it doesn't require access to a graphical environment, and running with windows disabled is usually somewhat faster.

-gui

Start Simics in command line mode but allow the GUI to open windows.

-no-gui

Start Simics in command line mode. No GUI windows will open except if explicitly requested using CLI commands.

-no-log

Turns off logging of issued CLI commands to the log file. This logging is deprecated and scheduled for future removal.

-log

Turns on logging of issued CLI commands to the `~/.simics/4.6/log` file on Unix and `<APPDATA>/Simics/4.6/log` on Windows (or any other file specified by `-log-file`). Logging is on by default but may have been disabled in the user preferences. This logging is deprecated and scheduled for future removal.

-log-file FILE

Log issued CLI commands to a different file than the default.

-v, --version, --v-short

Print the Simics version number and exit. Other information printed include the compilers used and the compile-time options applied to this specific build. -v-short only prints out Simics version.

-license-file FILE

Specify the FlexNet license file to use. This will make Simics ignore the preference setting and the `SIMICS_LICENSE_FILE` environment variable.

-readme

Print the Simics README information and exit.

-license

Print the Simics license text and exit. This is the default license and is also included in the LICENSE file.

-batch-mode

Run in batch mode. This means that Simics will exit when all commands and scripts given on the command line have executed, or as soon as an error occurs. When an error has occurred, Simics will immediately exit with a non-zero status. If all commands run to completion, Simics exits with status 0, indicating success. Simics will not use the command history file in batch mode.

-verbose

Make Simics more verbose. This makes Simics give ample details about the execution. This is the opposite of the -quiet flag. The quiet and verbose flags turn off each other. The default is somewhere in the middle with a “reasonable” output level.

Setting the verbose flag can be useful to figure out problems that you may be having with Simics. Some friendly warnings are not printed unless verbose is turned on.

Note that there will be quite a bit of information printed, most of which isn’t usually needed.

-fast

“Fast” mode is the default mode of execution. This option can be used to override simulation mode set in preferences.

-stall

Start the stall version of Simics if available. For more information about this version and its limitations, see the *Simulation Modes* section in *Hindsight User’s Guide*.

-q or -quiet

Make Simics less verbose. This makes Simics output less information, which is useful for batch execution. The opposite effect can be obtained with the -verbose flag. The quiet and verbose flags turn off each other. If you want Simics to be really quiet you can also give the -no-copyright flag.

Note that Simics is rather quiet by default, so this flag usually doesn't do all that much.

-c FILE

Load a configuration file. This is equivalent to issuing a **read-configuration** command after Simics has started.

If this flag is the last on the command line, **-c** may be omitted.

-x FILE

Run commands from a script file. The script file is usually named with a `.simics` suffix.

If this flag is the last on the command line, **-x** may be omitted.

-p FILE

Run code from a Python file.

-e COMMAND

Execute a CLI command. Equivalent of typing in the command at the command line prompt.

Advanced Options

-no-settings

Do not read any settings files, such as preferences.

-workspace PATH

Specify the workspace directory to run Simics in.

-wdeprecated

Warn all use of deprecated features.

-wall-deprecated

Warn the use of features that *may* be deprecated in the future. Note that this may not be correct since the actual API is not finalized until the actual release, and Simics itself may still use such features.

-no-wdeprecated

Turn off warnings about deprecated features.

-wmultithread

Warn about non-thread-safe behavior even if Simics is running single-threaded.

-no-wmultithread

Turn off warnings about non-thread-safe behavior (only if Simics is running single-threaded).

-E, -expire TIME

Check out licenses with an expire time, for off-line use. Valid formats are:

- dd-mmm-yyyy[:hh:mm] – where:

dd

day number in month

mmm

abbreviated name of month

yyyy
 year including century

hh
 hour in 24-hour format

mm
 minutes.

- +<num>h – setting the license to expire <num> hours from now.
- +<num>d – setting the license to expire <num> days from now.

-L
 Add a path to the directory list that Simics searches for modules.

-python-verbose
 Set the Python verbose flag. This will, for example, make Python print information when Python modules are being loaded. This is the same option as the **-v** flag in mini-python.

-werror
 Treat many warnings in Simics as errors and exit.

-no-werror
 Do not treat warnings in Simics as errors. This is the default behavior but may have been disabled in the user preferences.

-no-stc
 Disable the Simics internal caches for memory operations and instruction fetches (STCs) that Simics uses to speed up the simulation.
 Disabling the STCs (this flag disables both the D-STC and the I-STC) can be useful for debugging plug-ins such as cache models, since it will ensure that all memory accesses are fully visible.
 It is possible to selectively disable the data and instruction STCs using the **-no-dstc** and **-no-istc** flags, respectively. The STCs are typically enabled by default, although in some versions of Simics they may be disabled. See also the **-stc** flag.

-stc
 Force the I-STC and D-STC to be enabled (default). See the **-no-stc** flag for more information. The STCs are typically enabled by default, although in some versions of Simics they may be disabled.

-no-istc
 Force the I-STC to be disabled. See the **-no-stc** flag for more information.

-no-dstc
 Force the D-STC to be disabled. See the **-no-stc** flag for more information.

-istc
 Force the I-STC to be enabled. See the **-no-stc** flag for more information.

-dstc
 Force the D-STC to be enabled. See the **-no-stc** flag for more information.

-no-copyright
 Do not display copyright information when starting Simics. See also the **-verbose** and **-quiet** flags.

-central ADDRESS : [PORT]

This flag is deprecated and will be removed in a future Simics version.

Connect to Simics Central. Simics Central may be running on another host, and this argument to this flag is an IP address and an optional port number where Simics Central is listening. The port number defaults to 1909 if omitted.

If the local host's address is supplied, and the port number is left out, Simics will try to connect to the default Unix file socket (`/tmp/simics-central`).

Also see the `-central :FILE` flag.

-central :FILE

This flag is deprecated and will be removed in a future Simics version.

Connect to Simics Central using a Unix file socket. This variant of the `-central` flag lets you specify a socket file.

-n

Do not run the commands from the `startup-commands` script files.

-echo

Enable command echoing. When echoing is enabled, Simics will echo all commands executed by startup scripts. Note that this only affects any startup scripts loaded after the `-echo` flag on the command line.

-core

Allow Simics to dump core on fatal signals.

-optional-central

This flag is deprecated and will be removed in a future Simics version.

Only use remote Simics Central if set in configuration.

Environment Variables

SIMICS_HOST

Overrides the host type detected by Simics. The value must be the name of the directory containing the host-specific files of a Simics installation. Typically a string on the form arch-os, e.g., `linux32`.

2.2 **addon-manager**

Synopsis

```
addon-manager [options] [-u <path>] [-s <path>] [-d <path>]
```

Description

Configure Simics add-on packages to work with the Simics installation. When run without any option, this script will list the current add-on packages configuration. If any package listed is invalid (for example, if the package has been deleted), the script will propose to update the configuration to remove the invalid packages.

Options

-b, --batch

The script will act according to the command-line arguments without asking any questions.

-c, --create-empty

Create an empty package list in the workspace. This will cause the workspace to ignore any package associations of the main Simics installation. Only valid when operating on a workspace or together with the -f parameter.

-C, --copy-from-simics-base

Create a new package list in a workspace, by copying the package list from the main Simics installation. Only valid when operating on a workspace or together with the -f parameter. Roughly equivalent to -u <simics_dir>

-d, --deselect <path>

Remove the package in <path> from the add-on package list. Note that listed packages that do not exist anymore will be removed automatically whenever this script runs.

-f, --package-list

Specify the file containing the package list. This is an alternative to the default (storing the list in the main Simics installation) or running addon-manager in a workspace.

-s, --select <path>

Add the package in <path> to the add-on package list.

-u, --upgrade <path>

Re-use the add-on package list found as <file> or present in <path>. The current list is backed-up as with a .backup extension.

-v, --version

Print the version of the script and the major version of Simics that it can be used with.

-w, --workspace <path>

Handle the package associations of the workspace <path> instead of the base package. This can also be done by running the addon-manager script located in <path>/bin.

2.3 craff

Synopsis

```
craff [options] file ...
```

Description

The compressed random access file format is used by Simics's image module to handle sparse block data. The **craff** command line utility can be used to create and manipulate craff files.

The default operation is to merge the input files, each of which can be either in craff format or raw files (e.g., disk or memory dumps) and to output the merged result to an

output file. The order of the input files is important: data from later input files override earlier files. Since craff files can have holes (gaps) in them, data from earlier files can propagate to the result.

Options

-o, --output=FILE

Specifies the output file name. Without this option, output will go to `craff.out`. A hyphen (-) means standard output, and implies -q. Writing to standard output is only possible in raw format (-d) or when writing a content map (-l).

-d, --decompress

Causes the output to be an ordinary (raw) file. The default is to output in the craff format. When this option is given, any holes in the output are filled with zero bytes.

-l, --content-map

Writes a human-readable map of the input files to the output. Blocks are marked D where data is present, and . (dot) where not.

-D, --diff

Instead of merging the input files, output the difference between the input files as a craff file. There must be exactly two input files. The output is the file that, when merged with the first file, would result in the second file. The input files may be in craff or raw format.

-z, --zero-fill-gaps

Fills gaps with zero bytes and marks them as present. Without this option, a gap in all input files will remain a gap in the output file.

-Z, --omit-zero-blocks

Treats blocks that consist entirely of zero bytes as empty in the output (i.e., they will be output as holes).

-b, --block-size=SIZE

Specifies the output block size in bytes. The size must be a power of 2, and may be specified in kilobytes using 'k' as a suffix. Without this option, the block size will be the smallest of the input files' block sizes.

This is the smallest unit of storage in a craff file. Larger values compress better, but will waste more space if only part of the block is present. A block size of at least 4k is recommended.

-s, --sub-block-size=SIZE

Specifies the output sub-block size in bytes. The size must be a power of 2, and may be specified in kilobytes using 'k' as a suffix. Without this option, the block size will be the smallest of the input files' sub-block sizes.

This is the smallest unit of granularity in a craff file. Smaller values allow more fine-grained control of what data is present, but uses more space. For disk dumps, the sector size is useful (often 512 bytes); for memory dumps, use the page size (such as 4k or 8k).

-i, --dir-entries=N

Specifies the number of entries in each directory node. Must be a power of 2.

- c, --compression=COMPRESSION**
Specifies the compression type: none or zlib. No compression is faster but zlib makes the files smaller.
- e, --extract=START**
Extracts a block to the output file. The extracted block starts at file offset START and the size is given by --extract-block-size. The output is in raw format.
- w, --extract-block-size=SIZE**
Size of block to extract with --extract.
- n, --info**
Displays file information for each of the input files to the standard output.
- q, --quiet**
Suppresses the progress indicator.
- v, --version**
Displays version information of the **craff** utility itself.
- h, --help**
Displays a summary of the command line options.

2.4 install-simics

Synopsis

```
install-simics.pl
(or) install-simics.pl [OPTIONS] <package1> [<key1>] <package2> [<key2>] ...
```

Description

Installs and configure Simics and Simics add-on packages.

arguments

- <packageN>
Filename of a package to install.
- <keyN>
Key corresponding to <packageN>. The key can be omitted if it is already cached or if the package is not encrypted.

Options

- a, --autoselect**
If Simics and Simics add-on packages are installed at the same time, the script will automatically configure the new Simics installation to use these add-on packages.
- b, --batch**
The script will act according to the command-line arguments without asking any questions.

-h, --help

Display this help text.

-p <path>, --prefix <path>

Specify the directory in which to install the packages (defaults to the last directory used, or /opt/simics/simics-4.6/).

-s <path>, --select-in <path>

Configure automatically the Simics installation in <path> to use the Simics add-on packages being installed.

-u <path>, --upgrade-from <path>

Re-use the configuration of an existing Simics installation located in <path>.

-v, --version

Print the version of the script and the major version of Simics that it can be used with.

--leave-tmp-files

Do not delete the temporary files created during installation.

2.5 workspace-setup

Synopsis

```
workspace-setup [options] [workspace]
```

Description

Creates or updates a Simics workspace for user scripts and modules. If workspace directory is omitted, the current working directory is used.

Options

-h, --help

Show help message and exit.

-v, --version

Prints information about Simics (version, installation directory).

-q, --quiet

Cancel the verbose flag.

-v, --verbose

Print more information about the actions taken by the script.

-n, --dry-run

Execute normally, but do not change or create any files.

--ignore-existing-files

Prevent the script from warning that the future workspace directory is not empty.

--ignore-cygwin-warning

(Windows-only) Prevent the script from checking whether the workspace directory looks like a Cygwin path or not.

2.5. *workspace-setup*

--force

Force the workspace-setup script to create or update a workspace. Files that would be overwritten will be backed up automatically.

--check-workspace-version

Check the version of the workspace, and return 1 if it needs to be created or updated, 0 otherwise.

Chapter 3

Commands

3.1 Complete List

!	execute a shell command
!=	not equal
#	treat the line as a comment
\$	get the value of a CLI variable
%	read register by name, module or string formatting
&	bitwise AND operation
*	arithmetic multiplication
+	arithmetic addition, string and list concatenation
+=	set a CLI variable
-	arithmetic subtraction
-=	set a CLI variable
->	access object attribute
/	arithmetic division
:	access component object in a component
<	less than
<<	bitwise left shift
<=	less or equal
<accel-vga>.info	print information about the device
<accel-vga>.pci-header	<i>deprecated</i> — print PCI device header
<accel-vga>.print-pci-config-reg	print PCI configuration registers
<accel-vga>.redraw	Redraw display
<accel-vga>.refresh-rate	Set rate at which accel-vga updated display
<accel-vga>.status	print status of the device
<accel-vga>.text-dump	Print text contents of display
<accel-vga>.wait-for-string	Wait for substring in text mode
<address_profiler>.address-profile-data	linear map of address profiling data

<address_profiler>.address-profile-info	general info about an address profiler
<address_profiler>.address-profile-toplist	print toplist of address profiling data
<AM79C973>.info	print information about the device
<AM79C973>.pci-header	<i>deprecated</i> — print PCI device header
<AM79C973>.print-pci-config-reg	print PCI configuration registers
<AM79C973>.status	print status of the device
<AT24Cxx>.info	print information about the device
<AT24Cxx>.status	print status of the device
<attr-meter>.info	print information about the device
<attr-meter>.status	print status of the device
<base-trace-mem-hier>.start	control default tracer
<base-trace-mem-hier>.stop	stop default tracer
<base-trace-mem-hier>.trace-start	alias for <base-trace-mem-hier>.start
<base-trace-mem-hier>.trace-stop	alias for <base-trace-mem-hier>.stop
<BCM5703C>.connect	<i>deprecated</i> — connect to a simulated Ethernet link
<BCM5703C>.disconnect	<i>deprecated</i> — disconnect from simulated link
<BCM5703C>.info	print information about the device
<BCM5703C>.pci-header	<i>deprecated</i> — print PCI device header
<BCM5703C>.print-pci-config-reg	print PCI configuration registers
<BCM5703C>.status	print status of the device
<BCM5704C>.connect	<i>deprecated</i> — connect to a simulated Ethernet link
<BCM5704C>.disconnect	<i>deprecated</i> — disconnect from simulated link
<BCM5704C>.info	print information about the device
<BCM5704C>.pci-header	<i>deprecated</i> — print PCI device header
<BCM5704C>.print-pci-config-reg	print PCI configuration registers
<BCM5704C>.status	print status of the device
<bitmask-translator>.info	print information about the device
<bitmask-translator>.status	print status of the device
<breakpoint>.break	set breakpoint
<breakpoint>.tbreak	set temporary breakpoint on current processor
<cell-and-clocks>.info	print information about the device
<cell-and-clocks>.status	print status of the device
<cell>.cpu-switch-time	get/set the time quantum for a given cell
<cell>.info	print information about the device
<central-client>.connect	connect to Simics Central
<central-client>.disconnect	disconnect from Simics Central
<central-client>.info	print information about the device
<central-client>.links	list connected links
<central-server>.connections	list current connections
<central-server>.info	print information about the device
<central-server>.wait-for-clients	list current connections

<code><CL-PD6729>.info</code>	print information about the device
<code><CL-PD6729>.pci-header</code>	<i>deprecated</i> — print PCI device header
<code><CL-PD6729>.print-pci-config-reg</code>	print PCI configuration registers
<code><CL-PD6729>.status</code>	print status of the device
<code><clipboard-gateway>.info</code>	print information about the device
<code><clipboard-gateway>.status</code>	print status of the device
<code><clock>.info</code>	print information about the device
<code><clock>.status</code>	print status of the device
<code><component>.connect</code>	<i>deprecated</i> — connect components
<code><component>.debug-program</code>	<i>deprecated</i> — debug program
<code><component>.delete</code>	delete non-instantiated components
<code><component>.disconnect</code>	disconnect component connector
<code><component>.get-component-object</code>	get named object from components
<code><component>.get-connection</code>	return connection information
<code><component>.get-connector-list</code>	return list of connectors
<code><component>.get-processor-list</code>	return list of processors
<code><component>.info</code>	print information about the device
<code><component>.status</code>	print status of the device
<code><connector>.info</code>	print information about the device
<code><connector>.status</code>	print status of the device
<code><context>.fin</code>	alias for <code><context>.finish-function</code>
<code><context>.finish</code>	alias for <code><context>.finish-function</code>
<code><context>.finish-function</code>	finish the current function
<code><context>.info</code>	print information about the device
<code><context>.n</code>	alias for <code><context>.next-line</code>
<code><context>.next</code>	alias for <code><context>.next-line</code>
<code><context>.next-instruction</code>	run to the next instruction, skipping subroutine calls
<code><context>.next-line</code>	run to the next source line, skipping subroutine calls
<code><context>.nexti</code>	alias for <code><context>.next-instruction</code>
<code><context>.ni</code>	alias for <code><context>.next-instruction</code>
<code><context>.off</code>	switch off context object
<code><context>.on</code>	switch on context object
<code><context>.reverse-next-instruction</code>	reverse to the previous instruction, skipping subroutine calls
<code><context>.reverse-next-line</code>	reverse to the previous source line, skipping subroutine calls
<code><context>.reverse-step-instruction</code>	reverse to the previous instruction
<code><context>.reverse-step-line</code>	reverse to the previous source line
<code><context>.reverse-until-activated</code>	reverse until context becomes active
<code><context>.reverse-until-active</code>	<i>deprecated</i> — reverse until context becomes active
<code><context>.reverse-until-deactivated</code>	reverse until context becomes inactive
<code><context>.rn</code>	alias for <code><context>.reverse-next-line</code>
<code><context>.rnext</code>	alias for <code><context>.reverse-next-line</code>

<code><context>.rnexti</code>	alias for <code><context>.reverse-next-instruction</code>
<code><context>.rni</code>	alias for <code><context>.reverse-next-instruction</code>
<code><context>.rs</code>	alias for <code><context>.reverse-step-line</code>
<code><context>.rsi</code>	alias for <code><context>.reverse-step-instruction</code>
<code><context>.rstep</code>	alias for <code><context>.reverse-step-line</code>
<code><context>.rstepi</code>	alias for <code><context>.reverse-step-instruction</code>
<code><context>.run-until-activated</code>	run until context becomes active
<code><context>.run-until-active</code>	<i>deprecated</i> — run until context becomes active
<code><context>.run-until-deactivated</code>	run until context becomes inactive
<code><context>.s</code>	alias for <code><context>.step-line</code>
<code><context>.si</code>	alias for <code><context>.step-instruction</code>
<code><context>.status</code>	print status of the device
<code><context>.step</code>	alias for <code><context>.step-line</code>
<code><context>.step-instruction</code>	run to the next instruction
<code><context>.step-line</code>	run to the next source line
<code><context>.stepi</code>	alias for <code><context>.step-instruction</code>
<code><context>.symtable</code>	set the symbol table of a context
<code><context>.ui</code>	alias for <code><context>.reverse-step-instruction</code>
<code><context>.uncall</code>	alias for <code><context>.uncall-function</code>
<code><context>.uncall-function</code>	reverse to when the current function was called
<code><context>.unstep-instruction</code>	alias for <code><context>.reverse-step-instruction</code>
<code><coverage_profiler>.info</code>	print information about the device
<code><coverage_profiler>.save</code>	save the coverage profile
<code><coverage_profiler>.status</code>	print status of the device
<code><cp3_quad100tx>.info</code>	print information about the device
<code><cp3_quad100tx>.status</code>	print status of the device
<code><cpu-group>.info</code>	print information about the device
<code><cycle>.cb</code>	alias for <code><cycle>.cycle-break</code>
<code><cycle>.cba</code>	alias for <code><cycle>.cycle-break-absolute</code>
<code><cycle>.cycle-break</code>	set cycle breakpoint
<code><cycle>.cycle-break-absolute</code>	set absolute cycle breakpoint
<code><cycle>.print-time</code>	print number of steps and cycles executed
<code><cycle>.ptime</code>	alias for <code><cycle>.print-time</code>
<code><cycle>.wait-for-cycle</code>	wait until reaching cycle
<code><cycle>.wait-for-time</code>	wait until reaching a specified time
<code><data-profiler>.clear</code>	clear data profiler
<code><datagram_link>.info</code>	print information about the device
<code><datagram_link>.status</code>	print status of the device
<code><datagram_link_endpoint>.info</code>	print information about the device
<code><datagram_link_endpoint>.status</code>	print status of the device
<code><datagram_link_impl>.info</code>	print information about the device

<datagram_link_impl>.status	print status of the device
<ddr-memory-module>.info	print information about the device
<ddr-memory-module>.status	print status of the device
<ddr2-memory-module>.info	print information about the device
<ddr2-memory-module>.status	print status of the device
<ddr3-memory-module>.info	print information about the device
<ddr3-memory-module>.status	print status of the device
<DEC21041>.connect	<i>deprecated</i> — connect to a simulated Ethernet link
<DEC21041>.disconnect	<i>deprecated</i> — disconnect from simulated link
<DEC21041>.info	print information about the device
<DEC21041>.pci-header	<i>deprecated</i> — print PCI device header
<DEC21041>.print-pci-config-reg	print PCI configuration registers
<DEC21041>.status	print status of the device
<DEC21140A-dml>.pci-header	<i>deprecated</i> — print PCI device header
<DEC21140A-dml>.print-pci-config-reg	print PCI configuration registers
<DEC21140A>.connect	<i>deprecated</i> — connect to a simulated Ethernet link
<DEC21140A>.disconnect	<i>deprecated</i> — disconnect from simulated link
<DEC21140A>.info	print information about the device
<DEC21140A>.pci-header	<i>deprecated</i> — print PCI device header
<DEC21140A>.print-pci-config-reg	print PCI configuration registers
<DEC21140A>.status	print status of the device
<DEC21143>.connect	<i>deprecated</i> — connect to a simulated Ethernet link
<DEC21143>.disconnect	<i>deprecated</i> — disconnect from simulated link
<DEC21143>.info	print information about the device
<DEC21143>.pci-header	<i>deprecated</i> — print PCI device header
<DEC21143>.print-pci-config-reg	print PCI configuration registers
<DEC21143>.status	print status of the device
<dm9161>.info	print information about the device
<dm9161>.status	print status of the device
<dummy-component>.info	print information about the device
<dummy-component>.status	print status of the device
<dynamic_link_connector>.info	print information about the device
<dynamic_link_connector>.status	print status of the device
<empty-device-c>.info	print information about the device
<empty-device-c>.status	print status of the device
<empty-device-cc>.info	print information about the device
<empty-device-cc>.status	print status of the device
<etg>.info	print information about the device
<etg>.packet-rate	Set or display the packets per second rate
<etg>.packet-size	Set or display the packet size
<etg>.start	Start generating traffic

<code><etg>.status</code>	print status of the device
<code><etg>.stop</code>	Stop generating traffic
<code><etg_comp>.info</code>	print information about the device
<code><etg_comp>.status</code>	print status of the device
<code><etg_panel>.info</code>	print information about the device
<code><etg_panel>.status</code>	print status of the device
<code><eth-cable-link-endpoint>.info</code>	print information about the device
<code><eth-cable-link-endpoint>.status</code>	print status of the device
<code><eth-cable-link>.info</code>	print information about the device
<code><eth-cable-link>.status</code>	print status of the device
<code><eth-hub-link-endpoint>.info</code>	print information about the device
<code><eth-hub-link-endpoint>.status</code>	print status of the device
<code><eth-hub-link>.info</code>	print information about the device
<code><eth-hub-link>.status</code>	print status of the device
<code><eth-link-snoop-endpoint>.info</code>	print information about the device
<code><eth-link-snoop-endpoint>.status</code>	print status of the device
<code><eth-probe>.delete</code>	print status of the device
<code><eth-probe>.ethereal</code>	alias for <code><eth-probe>.wireshark</code>
<code><eth-probe>.ethereal-stop</code>	alias for <code><eth-probe>.wireshark-stop</code>
<code><eth-probe>.info</code>	print information about the device
<code><eth-probe>.pcap-dump</code>	Dump network traffic to a file, in libpcap format
<code><eth-probe>.pcap-dump-stop</code>	stop the current dump
<code><eth-probe>.tcpdump</code>	run the tcpdump program
<code><eth-probe>.tcpdump-stop</code>	stop the current tcpdump capture
<code><eth-probe>.wireshark</code>	run the wireshark/ethereal program
<code><eth-probe>.wireshark-stop</code>	stop the current wireshark capture
<code><eth-switch-link-endpoint>.info</code>	print information about the device
<code><eth-switch-link-endpoint>.status</code>	print status of the device
<code><eth-switch-link-snoop-endpoint>.info</code>	print information about the device
<code><eth-switch-link-snoop-endpoint>.status</code>	print status of the device
<code><eth-switch-link>.info</code>	print information about the device
<code><eth-switch-link>.status</code>	print status of the device
<code><eth-transceiver>.connect</code>	<i>deprecated</i> — connect to a simulated Ethernet link
<code><eth-transceiver>.disconnect</code>	<i>deprecated</i> — disconnect from simulated link
<code><eth-transceiver>.info</code>	print information about the device
<code><eth-transceiver>.status</code>	print status of the device
<code><eth_injector>.info</code>	print information about the device
<code><eth_injector>.start</code>	Start pcap playback
<code><eth_injector>.status</code>	print status of the device
<code><eth_injector_comp>.info</code>	print information about the device
<code><eth_injector_comp>.status</code>	print status of the device

<code><ethernet-link>.connect-real-network-bridge</code>	connect to the real network
<code><ethernet-link>.connect-real-network-host</code>	connect to the real network
<code><ethernet-link>.connect-real-network-napt</code>	enable NAPT from simulated network <i>deprecated</i> — connect to the real network
<code><ethernet-link>.connect-real-network-router</code>	disconnect from the real network
<code><ethernet-link>.disconnect-real-network</code>	print information about the device
<code><ethernet-link>.info</code>	print status of the device
<code><ethernet-link>.status</code>	connect to the real network
<code><ethernet_cable>.connect-real-network-bridge</code>	connect to the real network
<code><ethernet_cable>.connect-real-network-host</code>	enable NAPT from simulated network <i>deprecated</i> — connect to the real network
<code><ethernet_cable>.connect-real-network-napt</code>	disconnect from the real network
<code><ethernet_cable>.connect-real-network-router</code>	alias for <code><ethernet_cable>.wireshark</code>
<code><ethernet_cable>.disconnect-real-network</code>	alias for <code><ethernet_cable>.wireshark-stop</code>
<code><ethernet_cable>.ethereal</code>	return the name of an unused connector
<code><ethernet_cable>.ethereal-stop</code>	print information about the device
<code><ethernet_cable>.get-free-connector</code>	Dump network traffic to a file, in libpcap format
<code><ethernet_cable>.info</code>	stop the current dump
<code><ethernet_cable>.pcap-dump</code>	Set the link's goal latency (in seconds).
<code><ethernet_cable>.pcap-dump-stop</code>	print status of the device
<code><ethernet_cable>.set-goal-latency</code>	run the tcpdump program
<code><ethernet_cable>.status</code>	stop the current tcpdump capture
<code><ethernet_cable>.tcpdump</code>	run the wireshark/ethereal program
<code><ethernet_cable>.tcpdump-stop</code>	stop the current wireshark capture
<code><ethernet_cable>.wireshark</code>	connect to the real network
<code><ethernet_cable>.wireshark-stop</code>	connect to the real network
<code><ethernet_hub>.connect-real-network-bridge</code>	enable NAPT from simulated network <i>deprecated</i> — connect to the real network
<code><ethernet_hub>.connect-real-network-host</code>	disconnect from the real network
<code><ethernet_hub>.connect-real-network-napt</code>	alias for <code><ethernet_hub>.wireshark</code>
<code><ethernet_hub>.connect-real-network-router</code>	alias for <code><ethernet_hub>.wireshark-stop</code>
<code><ethernet_hub>.disconnect-real-network</code>	return the name of an unused connector
<code><ethernet_hub>.ethereal</code>	print information about the device
<code><ethernet_hub>.ethereal-stop</code>	Dump network traffic to a file, in libpcap format
<code><ethernet_hub>.get-free-connector</code>	stop the current dump
<code><ethernet_hub>.info</code>	Set the link's goal latency (in seconds).
<code><ethernet_hub>.pcap-dump</code>	print status of the device
<code><ethernet_hub>.pcap-dump-stop</code>	run the tcpdump program
<code><ethernet_hub>.set-goal-latency</code>	stop the current tcpdump capture
<code><ethernet_hub>.status</code>	run the wireshark/ethereal program
<code><ethernet_hub>.tcpdump</code>	stop the current wireshark capture
<code><ethernet_hub>.tcpdump-stop</code>	
<code><ethernet_hub>.wireshark</code>	
<code><ethernet_hub>.wireshark-stop</code>	

3.1. Complete List

<ethernet_switch>.connect-real-network-bridge	connect to the real network
<ethernet_switch>.connect-real-network-host	connect to the real network
<ethernet_switch>.connect-real-network-napt	enable NAPT from simulated network <i>deprecated</i> — connect to the real network
<ethernet_switch>.connect-real-network-router	disconnect from the real network
<ethernet_switch>.disconnect-real-network	alias for <ethernet_switch>.wireshark
<ethernet_switch>.ethereal	alias for <ethernet_switch>.wireshark-stop
<ethernet_switch>.ethereal-stop	return the name of an unused connector
<ethernet_switch>.get-free-connector	print information about the device
<ethernet_switch>.info	Dump network traffic to a file, in libpcap format
<ethernet_switch>.pcap-dump	stop the current dump
<ethernet_switch>.pcap-dump-stop	Set the link's goal latency (in seconds).
<ethernet_switch>.set-goal-latency	print status of the device
<ethernet_switch>.status	run the tcpdump program
<ethernet_switch>.tcpdump	stop the current tcpdump capture
<ethernet_switch>.tcpdump-stop	run the wireshark/ethereal program
<ethernet_switch>.wireshark	stop the current wireshark capture
<ethernet_switch>.wireshark-stop	add a VLAN definition and corresponding connection
<ethernet_vlan_switch>.add-vlan	enable NAPT from simulated network
<ethernet_vlan_switch>.connect-real-network-napt	alias for <ethernet_vlan_switch>.wireshark
<ethernet_vlan_switch>.ethereal	alias for <ethernet_vlan_switch>.wireshark-stop
<ethernet_vlan_switch>.ethereal-stop	return the name of an unused connector
<ethernet_vlan_switch>.get-free-connector	return the name of an unused trunk connector
<ethernet_vlan_switch>.get-free-trunk-connector	print information about the device
<ethernet_vlan_switch>.info	Dump network traffic to a file, in libpcap format
<ethernet_vlan_switch>.pcap-dump	stop the current dump
<ethernet_vlan_switch>.pcap-dump-stop	Set the link's goal latency (in seconds).
<ethernet_vlan_switch>.set-goal-latency	print status of the device
<ethernet_vlan_switch>.status	run the tcpdump program
<ethernet_vlan_switch>.tcpdump	stop the current tcpdump capture
<ethernet_vlan_switch>.tcpdump-stop	run the wireshark/ethereal program
<ethernet_vlan_switch>.wireshark	stop the current wireshark capture
<ethernet_vlan_switch>.wireshark-stop	print information about the device
<example-keypad>.info	print status of the device
<example-keypad>.status	print information about the device
<example-status-panel>.info	print status of the device
<example-status-panel>.status	print status of the device
<fc-disk>.add-diff-file	add a diff file to the image
<fc-disk>.add-diff-partial-file	add a partial diff file to the image
<fc-disk>.add-sun-partition	add partition from a file
<fc-disk>.create-sun-vtoc-header	write a new VTOC to a Sun disk
<fc-disk>.create-sun-vtoc-partition	write partition data in the VTOC on a Sun disk

<fc-disk>.delete-sun-vtoc-partition	delete partition data from the VTOC on a Sun disk
<fc-disk>.dump-sun-partition	write partition as a file
<fc-disk>.info	information about current state of the fibre-channel disk
<fc-disk>.print-sun-vtoc	print the VTOC for a Sun disk
<fc-disk>.save-diff-file	save diff file to disk
<file-cdrom>.delete	delete an unused file-cdrom object
<firewire_bus>.info	print information about the device
<firewire_bus>.status	print status of the device
<firewire_sample_device>.info	print information about the device
<firewire_sample_device>.status	print status of the device
<frequency_bus>.info	print information about the device
<frequency_bus>.status	print status of the device
<ftp-control>.info	print information about the device
<ftp-control>.status	print status of the device
<ftp-data>.info	print information about the device
<ftp-data>.status	print status of the device
<ftp-service>.info	print information about the device
<ftp-service>.status	print status of the device
<g-cache>.add-profiler	Add a profiler to the cache
<g-cache>.info	print the cache information
<g-cache>.remove-profiler	Remove a profiler from the cache
<g-cache>.reset-cache-lines	Reset all the cache lines
<g-cache>.reset-statistics	reset the cache statistics
<g-cache>.statistics	print the cache statistics
<g-cache>.status	print the cache lines status
<gdb-remote>.disconnect	disconnect from the remote gdb
<gdb-remote>.follow-context	follow context
<gdb-remote>.info	print information about the device
<gdb-remote>.signal	tell remote gdb we got a signal
<gdb-remote>.target	set target CPU for gdb connection
<generic-flash-memory>.accept-inquiries	<i>deprecated</i> — set whether or not to handle inquiry accesses
<generic-flash-memory>.info	print information about the device
<generic-flash-memory>.status	print status of the device
<generic mmc-card>.info	print information about the device
<generic mmc-card>.status	print status of the device
<generic_eth_phy>.info	print information about the device
<generic_eth_phy>.status	print status of the device
<generic_pcie_switch>.info	print information about the device
<generic_pcie_switch>.status	print status of the device
<generic_pcie_switch_port>.info	print information about the device
<generic_pcie_switch_port>.pci-header	<i>deprecated</i> — print PCI device header

<generic_pcie_switch_port>.print-pci-config-reg	print PCI configuration registers
<generic_pcie_switch_port>.status	print status of the device
<generic_spi_flash>.info	print information about the device
<generic_spi_flash>.status	print status of the device
<gfx-console>.auto-release	get/set auto-release flag
<gfx-console>.break	break on a graphics event specified by filename
<gfx-console>.close	close console window
<gfx-console>.delete	delete breakpoint
<gfx-console>.disable-input	ignore console input
<gfx-console>.disable-visual-feedback	disable visual feedback
<gfx-console>.enable-input	enable console input
<gfx-console>.enable-visual-feedback	enable visual feedback
<gfx-console>.grab-setup	set grab button and modifier
<gfx-console>.input	send string to a console
<gfx-console>.open	open console window
<gfx-console>.refresh	refresh console
<gfx-console>.save-bmp	save BMP image
<gfx-console>.save-break-xy	specify and save a graphical breakpoint
<gfx-console>.save-png	save PNG image
<gfx-console>.switch-to-text-console	replace graphics console with text console
<hap-meter>.info	print information about the device
<hap-meter>.listen-for-exceptions	listen for exception haps
<hap-meter>.listen-for-hap	listen for a specified hap
<hap-meter>.status	print status of the device
<host-serial-console>.switch-to-serial-link	<i>deprecated</i> —
<host-serial-console>.switch-to-telnet-console	<i>deprecated</i> —
<host-serial-console>.switch-to-text-console	<i>deprecated</i> —
<hostfs>.info	print information about the device
<hostfs>.root	get or set the hostfs root directory
<hostfs>.status	print status of the device
<hypersim-pattern-matcher>.delete-pattern	Remove a pattern from the matcher
<hypersim-pattern-matcher>.info	print information about the device
<hypersim-pattern-matcher>.status	print status of the device
<i21150>.info	print information about the device
<i21150>.pci-header	<i>deprecated</i> — print PCI device header
<i21150>.print-pci-config-reg	print PCI configuration registers
<i21150>.status	print status of the device
<i21152>.info	print information about the device
<i21152>.pci-header	<i>deprecated</i> — print PCI device header
<i21152>.print-pci-config-reg	print PCI configuration registers
<i21152>.status	print status of the device

<code><i21154>.info</code>	print information about the device
<code><i21154>.pci-header</code>	<i>deprecated</i> — print PCI device header
<code><i21154>.print-pci-config-reg</code>	print PCI configuration registers
<code><i21154>.status</code>	print status of the device
<code><i2c-bus>.info</code>	print information about the device
<code><i2c-bus>.status</code>	print status of the device
<code><i2c-link-endpoint>.info</code>	print information about the device
<code><i2c-link-endpoint>.status</code>	print status of the device
<code><i2c-link-impl>.info</code>	print information about the device
<code><i2c-link-impl>.status</code>	print status of the device
<code><i2c_link_v1>.info</code>	print information about the device
<code><i2c_link_v1>.status</code>	print status of the device
<code><i2c_link_v2>.info</code>	print information about the device
<code><i2c_link_v2>.status</code>	print status of the device
<code><i2c_slave_v2_to_bus_adapter>.info</code>	print information about the device
<code><i2c_slave_v2_to_bus_adapter>.status</code>	print status of the device
<code><i2c_wire>.info</code>	print information about the device
<code><i2c_wire>.status</code>	print status of the device
<code><i82543>.connect</code>	<i>deprecated</i> — connect to a simulated Ethernet link
<code><i82543>.disconnect</code>	<i>deprecated</i> — disconnect from simulated link
<code><i82543>.info</code>	print information about the device
<code><i82543>.pci-header</code>	<i>deprecated</i> — print PCI device header
<code><i82543>.print-pci-config-reg</code>	print PCI configuration registers
<code><i82543>.status</code>	print status of the device
<code><i82546>.connect</code>	<i>deprecated</i> — connect to a simulated Ethernet link
<code><i82546>.disconnect</code>	<i>deprecated</i> — disconnect from simulated link
<code><i82546>.info</code>	print information about the device
<code><i82546>.pci-header</code>	<i>deprecated</i> — print PCI device header
<code><i82546>.print-pci-config-reg</code>	print PCI configuration registers
<code><i82546>.status</code>	print status of the device
<code><i82559>.info</code>	print information about the device
<code><i82559>.pci-header</code>	<i>deprecated</i> — print PCI device header
<code><i82559>.print-pci-config-reg</code>	print PCI configuration registers
<code><i82559>.status</code>	print status of the device
<code><ide-cdrom>.eject</code>	eject media from CD-ROM drive
<code><ide-cdrom>.info</code>	print information about the device
<code><ide-cdrom>.insert</code>	insert media in CD-ROM drive
<code><ide-disk>.add-diff-file</code>	add a diff file to the image
<code><ide-disk>.add-diff-partial-file</code>	add a partial diff file to the image
<code><ide-disk>.add-sun-partition</code>	add partition from a file
<code><ide-disk>.create-partition</code>	add a partition to disk

<ide-disk>.create-sun-vtoc-header	write a new VTOC to a Sun disk
<ide-disk>.create-sun-vtoc-partition	write partition data in the VTOC on a Sun disk
<ide-disk>.default-translation	get or set the default CHS translation
<ide-disk>.delete-sun-vtoc-partition	delete partition data from the VTOC on a Sun disk
<ide-disk>.dump-sun-partition	write partition as a file
<ide-disk>.info	print information about the device
<ide-disk>.print-partition-info	print info about a pc-style partition
<ide-disk>.print-partition-table	print the disk's pc-style partition table
<ide-disk>.print-sun-vtoc	print the VTOC for a Sun disk
<ide-disk>.save-diff-file	save diff file to disk
<ide>.info	print information about the device
<ide>.status	print status of the device
<image>.add-diff-file	add a diff file to the image
<image>.add-partial-diff-file	add a partial diff file to the image
<image>.info	print information about the device
<image>.save	save image to disk
<image>.save-diff-file	save changes since last checkpoint
<image>.set	set bytes in image to specified value
<image>.status	print status of the device
<image>.x	examine image data
<instruction-data-splitter>.info	print information about the device
<instruction-data-splitter>.status	print status of the device
<int_register>.break-cr	break on control register updates
<int_register>.read-reg	read a register
<int_register>.register-number	<i>deprecated</i> — get the number of a processor register
<int_register>.trace-cr	trace control register updates
<int_register>.unbreak-cr	break on control register updates
<int_register>.untrace-cr	trace control register updates
<int_register>.wait-for-register-read	wait for a register read
<int_register>.wait-for-register-write	wait for a register write
<int_register>.write-reg	write to register
<isa-fourport>.info	print information about the device
<isa-fourport>.status	print status of the device
<isa-lance>.info	print information about the device
<isa-lance>.status	print status of the device
<isa-vga>.info	print information about the device
<isa-vga>.status	print status of the device
<ISP1040>.info	print information about the device
<ISP1040>.pci-header	<i>deprecated</i> — print PCI device header
<ISP1040>.print-pci-config-regs	print PCI configuration registers
<ISP1040>.status	print status of the device

<ISP2200>.info	print information about the device
<ISP2200>.pci-header	<i>deprecated</i> — print PCI device header
<ISP2200>.print-pci-config-reg	print PCI configuration registers
<ISP2200>.status	print status of the device
<mem-traffic-meter>.info	print information about the device
<mem-traffic-meter>.status	print status of the device
<memory-space>.add-map	map device in a memory-space
<memory-space>.del-map	remove device map from a memory-space
<memory-space>.get	get value of physical address without side-effects
<memory-space>.info	print information about the device
<memory-space>.load-binary	load binary (executable) file into memory
<memory-space>.load-file	load file into memory
<memory-space>.map	list memory map
<memory-space>.read	get value of physical address
<memory-space>.set	set physical address to specified value without side-effects
<memory-space>.status	print status of the device
<memory-space>.write	set physical address to specified value
<memory-space>.x	examine raw memory contents
<memory-timer>.info	print information about the device
<memory-timer>.status	print status of the device
<memory_space>.debug	get debug object
<micron_mtfc2ggqdi_emmc_card>.info	print information about the device
<micron_mtfc2ggqdi_emmc_card>.status	print status of the device
<micron_mtfc4ggqdi_emmc_card>.info	print information about the device
<micron_mtfc4ggqdi_emmc_card>.status	print status of the device
<micron_mtfc4ggqdi_sdhc_card>.info	print information about the device
<micron_mtfc4ggqdi_sdhc_card>.status	print status of the device
<mii-management-bus>.info	print information about the device
<mii-management-bus>.status	print status of the device
<mii-transceiver>.connect	<i>deprecated</i> — connect to a simulated Ethernet link
<mii-transceiver>.disconnect	<i>deprecated</i> — disconnect from simulated link
<mii-transceiver>.info	print information about the device
<mii-transceiver>.status	print status of the device
<ms1553-link>.capture-start	start traffic recorder
<ms1553-link>.capture-stop	stop traffic recorder
<ms1553-link>.info	print information about the device
<ms1553-link>.playback-start	start traffic generator
<ms1553-link>.playback-stop	stop traffic generation
<ms1553-link>.status	print status of the device
<mtprof>.cellstat	display cell profiling information
<mtprof>.disable	disable multithreaded simulation profiling

3.1. Complete List

<mtprof>.enable	enable multithreaded simulation profiling
<mtprof>.info	print information about the device
<mtprof>.modelstat	display ideal execution time on a sufficiently parallel hardware
<mtprof>.save-data	save profiling data to file
<mtprof>.status	print status of the device
<NS16450>.info	print information about the device
<NS16450>.status	print status of the device
<NS16550>.info	print information about the device
<NS16550>.status	print status of the device
<NS16550_c>.info	print information about the device
<NS16550_c>.status	print status of the device
<onfi_flash>.info	print information about the device
<onfi_flash>.status	print status of the device
<os_awareness>.active-node	list the active nodes
<os_awareness>.break-enter	break the simulation when a processor becomes active
<os_awareness>.break-exit	break the simulation when a processor becomes active
<os_awareness>.code-coverage	collect a coverage profile
<os_awareness>.disable-tracker	stop using software tracking
<os_awareness>.enable-tracker	enable software tracking
<os_awareness>.find	find a node
<os_awareness>.info	print information about the device
<os_awareness>.linux-autodetect-settings	Try to autodetect settings for the Linux tracker
<os_awareness>.list	list processes/tasks
<os_awareness>.load-parameters	Load tracker parameter settings
<os_awareness>.log-syscalls	start logging system calls
<os_awareness>.node-info	show software tracker node information
<os_awareness>.node-tree	list software node tree
<os_awareness>.ose-detect-settings	detect settings for the OSE tracker
<os_awareness>.partition-settings	configure the partition tracker
<os_awareness>.qnx-detect-settings	detect settings for the QNX tracker
<os_awareness>.status	print status of the device
<os_awareness>.track	let a context track a process, thread, task, etc.
<os_awareness>.unbreak	cancel a software breakpoint
<os_awareness>.vxworks-detect-settings	detect settings for the VxWorks tracker
<os_awareness>.wait-for-activated	Wait for a processor to become activated on the node <i>deprecated</i> —
<os_awareness>.wait-for-active	Wait for a processor to become deactivated on the node <i>deprecated</i> —
<os_awareness>.wait-for-deactivated	detect settings for the Wind River hypervisor tracker
<os_awareness>.wait-for-inactive	print information about the device
<os_awareness>.wr-hypervisor-detect-settings	print status of the device
<pc-dual-serial-ports>.info	
<pc-dual-serial-ports>.status	

<pc-floppy-controller>.info	print information about the device
<pc-floppy-controller>.status	print status of the device
<pc-quad-serial-ports>.info	print information about the device
<pc-quad-serial-ports>.status	print status of the device
<pc-single-parallel-port>.info	print information about the device
<pc-single-parallel-port>.status	print status of the device
<pci-accel-vga>.info	print information about the device
<pci-accel-vga>.status	print status of the device
<pci-am79c973>.info	print information about the device
<pci-am79c973>.status	print status of the device
<pci-bcm5703c>.info	print information about the device
<pci-bcm5703c>.status	print status of the device
<pci-bcm5704c>.info	print information about the device
<pci-bcm5704c>.status	print status of the device
<pci-bus>.info	print information about the device
<pci-dec21041>.info	print information about the device
<pci-dec21041>.status	print status of the device
<pci-dec21140a-dml>.info	print information about the device
<pci-dec21140a-dml>.status	print status of the device
<pci-dec21140a>.info	print information about the device
<pci-dec21140a>.status	print status of the device
<pci-dec21143>.info	print information about the device
<pci-dec21143>.status	print status of the device
<pci-i21152>.info	print information about the device
<pci-i21152>.status	print status of the device
<pci-i82543gc>.info	print information about the device
<pci-i82543gc>.status	print status of the device
<pci-i82546bg>.info	print information about the device
<pci-i82546bg>.status	print status of the device
<pci-i82559>.info	print information about the device
<pci-i82559>.status	print status of the device
<pci-ispl040>.info	print information about the device
<pci-ispl040>.status	print status of the device
<pci-ispl2200>.info	print information about the device
<pci-ispl2200>.status	print status of the device
<pci-pd6729>.info	print information about the device
<pci-pd6729>.status	print status of the device
<pci-pmc1553>.info	print information about the device
<pci-pmc1553>.status	print status of the device
<pci-sil680a>.info	print information about the device
<pci-sil680a>.status	print status of the device

<pci-sym53c810>.info	print information about the device
<pci-sym53c810>.status	print status of the device
<pci-sym53c875>.info	print information about the device
<pci-sym53c875>.status	print status of the device
<pci-sym53c876>.info	print information about the device
<pci-sym53c876>.status	print status of the device
<pci-tsb12lv26>.info	print information about the device
<pci-tsb12lv26>.status	print status of the device
<pci-vga>.info	print information about the device
<pci-vga>.status	print status of the device
<pcie-bus>.info	print information about the device
<persistent-ram>.info	print information about the device
<persistent-ram>.status	print status of the device
<phy-mii-transceiver>.info	print information about the device
<phy-mii-transceiver>.status	print status of the device
<phy_comp>.info	print information about the device
<phy_comp>.status	print status of the device
<PMC1553>.hard-reset	hard reset
<PMC1553>.info	print information about the device
<PMC1553>.pci-header	<i>deprecated</i> — print PCI device header
<PMC1553>.print-pci-config-reg	print PCI configuration registers
<PMC1553>.soft-reset	soft reset
<PMC1553>.status	print status of the device
<port-forward-outgoing-server>.status	print status of the device
<port-space>.add-map	map device in a port-space
<port-space>.del-map	remove device map from a port-space
<port-space>.get	get value of physical address without side-effects
<port-space>.info	print information about the device
<port-space>.map	list port map
<port-space>.read	get value of physical address
<port-space>.set	set physical address to specified value without side-effects
<port-space>.status	print status of the device
<port-space>.write	set physical address to specified value
<preferences>.info	print information about the device
<preferences>.status	print status of the device
<ps2-keyboard-mouse>.info	print information about the device
<ps2-keyboard-mouse>.status	print status of the device
<ram>.info	print information about the device
<ram>.status	print status of the device
<rapidio_link>.info	print information about the device
<rapidio_link>.status	print status of the device

<rapidio_link_endpoint>.info	print information about the device
<rapidio_link_endpoint>.status	print status of the device
<rapidio_link_impl>.info	print information about the device
<rapidio_link_impl>.status	print status of the device
<rapidio_tape>.info	print information about the device
<rapidio_tape>.playback-stop	stop traffic generation
<rapidio_tape>.recorder-stop	stop traffic recording
<rapidio_tape>.status	print status of the device
<real-network-bridge>.disconnect-real-network	disconnect from the real network
<real-network-bridge>.finish-connection	Finalize a real network connection
<real-network-bridge>.info	print information about the device
<real-network-bridge>.status	print status of the device
<real-network-host>.disconnect-real-network	disconnect from the real network
<real-network-host>.finish-connection	Finalize a real network connection
<real-network-host>.info	print information about the device
<real-network-host>.status	print status of the device
<real-network-router>.disconnect-real-network	disconnect from the real network
<real-network-router>.info	print information about the device
<real-network-router>.status	print status of the device
<realtime>.disable	disable real-time behavior
<realtime>.enable	enable real-time behavior
<realtime>.info	print information about the device
<realtime>.status	print status of the device
<recorder>.info	print information about the device
<recorder>.playback-start	play back recorded I/O
<recorder>.playback-stop	stop playback
<recorder>.recorder-save	save recorder
<recorder>.recorder-start	record input to file
<recorder>.recorder-stop	stop recorder
<recorder>.status	print status of the device
<remote_sync_domain>.info	print information about the device
<remote_sync_domain>.status	print status of the device
<remote_sync_node>.info	print information about the device
<remote_sync_node>.status	print status of the device
<remote_sync_server>.info	print information about the device
<remote_sync_server>.status	print status of the device
<rn-eth-bridge-raw>.info	print information about the device
<rn-eth-bridge-raw>.status	print status of the device
<rn-eth-bridge-tap>.finish-connection	Finalize a real network connection
<rn-eth-bridge-tap>.info	print information about the device
<rn-eth-bridge-tap>.status	print status of the device

<rn-eth-proxy-raw>.info	print information about the device
<rn-eth-proxy-raw>.status	print status of the device
<rn-ip-router-raw>.info	print information about the device
<rn-ip-router-raw>.status	print status of the device
<rom>.info	print information about the device
<rom>.status	print status of the device
<sample-gcache>.info	print information about the device
<sample-gcache>.status	print status of the device
<scsi-bus>.info	print information about the device
<scsi-bus>.status	print status of the device
<scsi-cdrom>.eject	eject media from CD-ROM drive
<scsi-cdrom>.info	print information about the device
<scsi-cdrom>.insert	insert medium in CD-ROM drive
<scsi-cdrom>.status	print status of the device
<scsi-disk>.add-diff-file	add a diff file to the image
<scsi-disk>.add-diff-partial-file	add a partial diff file to the image
<scsi-disk>.add-sun-partition	add partition from a file
<scsi-disk>.create-partition	add a partition to disk
<scsi-disk>.create-sun-vtoc-header	write a new VTOC to a Sun disk
<scsi-disk>.create-sun-vtoc-partition	write partition data in the VTOC on a Sun disk
<scsi-disk>.delete-sun-vtoc-partition	delete partition data from the VTOC on a Sun disk
<scsi-disk>.dump-sun-partition	write partition as a file
<scsi-disk>.info	print information about the device
<scsi-disk>.print-partition-info	print info about a pc-style partition
<scsi-disk>.print-partition-table	print the disk's pc-style partition table
<scsi-disk>.print-sun-vtoc	print the VTOC for a Sun disk
<scsi-disk>.save-diff-file	save diff file to disk
<scsi-disk>.status	print status of the device
<sDRAM-memory-module>.info	print information about the device
<sDRAM-memory-module>.status	print status of the device
<selfprof>.info	print information about the device
<selfprof>.status	print status of the device
<ser-link-endpoint>.info	print information about the device
<ser-link-endpoint>.status	print status of the device
<ser-link-impl>.info	print information about the device
<ser-link-impl>.status	print status of the device
<ser_link>.info	print information about the device
<ser_link>.status	print status of the device
<serial-link>.info	print information about the device
<serial-link>.status	print status of the device
<service-node-device>.info	print information about the device

<service-node-device>.status	print status of the device
<service-node>.add-host	add host entry
<service-node>.arp	inspect and manipulate ARP table
<service-node>.connect	<i>deprecated</i> — connect to an ethernet link
<service-node>.delete-host	delete host entry
<service-node>.dhcp-add-pool	add DHCP pool
<service-node>.dhcp-leases	show DHCP leases
<service-node>.disable-real-dns	disable real DNS
<service-node>.disable-service	disable network service
<service-node>.enable-ftp-alg	enable FTP ALG
<service-node>.enable-real-dns	enable real DNS
<service-node>.enable-service	enable network service
<service-node>.info	print information about the device
<service-node>.list-host-info	print host info database
<service-node>.route	show the routing table
<service-node>.route-add	add an entry to the routing table
<service-node>.set-tftp-directory	set TFTP directory
<service-node>.status	print status of the device
<service-node>.tcpip-info	show TCP/IP info
<set-memory>.info	print information about the device
<set-memory>.status	print status of the device
<signal-bus>.info	print information about the device
<signal-bus>.status	print status of the device
<signal_link>.info	print information about the device
<signal_link>.status	print status of the device
<signal_link_endpoint>.info	print information about the device
<signal_link_endpoint>.status	print status of the device
<signal_link_impl>.info	print information about the device
<signal_link_impl>.status	print status of the device
<signal_to_interrupt>.info	print information about the device
<signal_to_interrupt>.status	print status of the device
<SIL680A>.info	print information about the device
<SIL680A>.pci-header	<i>deprecated</i> — print PCI device header
<SIL680A>.print-pci-config-reg	print PCI configuration registers
<SIL680A>.status	print status of the device
<sim>.info	print information about the device
<sim>.status	print status of the device
<simple-fc-disk>.info	print information about the device
<simple-fc-disk>.status	print status of the device
<simple-scsi-disk>.add-diff-file	add a diff file to the image
<simple-scsi-disk>.add-diff-partial-file	add a partial diff file to the image

<simple-scsi-disk>.add-sun-partition	add partition from a file
<simple-scsi-disk>.create-partition	add a partition to disk
<simple-scsi-disk>.create-sun-vtoc-header	write a new VTOC to a Sun disk
<simple-scsi-disk>.create-sun-vtoc-partition	write partition data in the VTOC on a Sun disk
<simple-scsi-disk>.delete-sun-vtoc-partition	delete partition data from the VTOC on a Sun disk
<simple-scsi-disk>.dump-sun-partition	write partition as a file
<simple-scsi-disk>.info	information about current state of the SCSI disk
<simple-scsi-disk>.print-partition-info	print info about a pc-style partition
<simple-scsi-disk>.print-partition-table	print the disk's pc-style partition table
<simple-scsi-disk>.print-sun-vtoc	print the VTOC for a Sun disk
<simple-scsi-disk>.save-diff-file	save diff file to disk
<simple_memory_module>.info	print information about the device
<simple_memory_module>.status	print status of the device
<sio-w83627hf>.info	print information about the device
<sio-w83627hf>.status	print status of the device
<slaver>.info	print information about the device
<slaver>.status	print status of the device
<software_tracker>.info	print information about the device
<software_tracker>.status	print status of the device
<state-assertion>.add	add an object to be asserted
<state-assertion>.add-mem-lis	add a memory listener on the specified memory space
<state-assertion>.fforward	fast-forward a state assertion file when comparing
<state-assertion>.info	provide information about the state assertion
<state-assertion>.start	start trace asserting/comparing
<state-assertion>.status	provide the status of the current state assertion
<state-assertion>.stop	stop trace asserting/comparing and close the file
<static_link_connector>.info	print information about the device
<static_link_connector>.status	print status of the device
<std-etg>.change-reference-clock	change the reference clock of an etg component
<std-etg>.info	print information about the device
<std-etg>.status	print status of the device
<std-ethernet-link>.info	print information about the device
<std-ethernet-link>.status	print status of the device
<std-firewire-bus>.info	print information about the device
<std-firewire-bus>.status	print status of the device
<std-firewire-sample-device>.info	print information about the device
<std-firewire-sample-device>.status	print status of the device
<std-generic-link-sample>.info	print information about the device
<std-generic-link-sample>.status	print status of the device
<std-generic-link>.info	print information about the device
<std-generic-link>.status	print status of the device

<std-graphics-console>.info	print information about the device
<std-graphics-console>.status	print status of the device
<std-graphics-console>.switch-to-text-console	replace the console with a text console
<std-host-serial-console>.info	print information about the device
<std-host-serial-console>.status	print status of the device
<std-host-serial-console>.switch-to-telnet-console	replace the console with a telnet console
<std-host-serial-console>.switch-to-text-console	replace the console with a text console
<std-ide-cdrom>.info	print information about the device
<std-ide-cdrom>.status	print status of the device
<std-ide-disk>.info	print information about the device
<std-ide-disk>.status	print status of the device
<std-ms1553-link>.info	print information about the device
<std-ms1553-link>.status	print status of the device
<std-pcmcia-flash-disk>.info	print information about the device
<std-pcmcia-flash-disk>.status	print status of the device
<std-scsi-bus>.info	print information about the device
<std-scsi-bus>.status	print status of the device
<std-scsi-cdrom>.info	print information about the device
<std-scsi-cdrom>.status	print status of the device
<std-scsi-disk>.info	print information about the device
<std-scsi-disk>.status	print status of the device
<std-serial-link>.info	print information about the device
<std-serial-link>.status	print status of the device
<std-server-console>.info	print information about the device
<std-server-console>.status	print status of the device
<std-service-node>.add-connector	add a service-node connector
<std-service-node>.add-host	add host entry
<std-service-node>.arp	inspect and manipulate ARP table
<std-service-node>.change-reference-clock	change the reference clock of a service-node component
<std-service-node>.connect-to-link	connect a service-node component to a link
<std-service-node>.delete-host	delete host entry
<std-service-node>.dhcp-add-pool	add DHCP pool
<std-service-node>.dhcp-leases	show DHCP leases
<std-service-node>.disable-real-dns	disable real DNS
<std-service-node>.disable-service	disable network service
<std-service-node>.enable-ftp-alg	enable FTP ALG
<std-service-node>.enable-real-dns	enable real DNS
<std-service-node>.enable-service	enable network service
<std-service-node>.info	print information about the device
<std-service-node>.list-host-info	print host info database
<std-service-node>.route	show the routing table

3.1. Complete List

<std-service-node>.route-add	add an entry to the routing table
<std-service-node>.set-tftp-directory	set TFTP directory
<std-service-node>.status	print status of the device
<std-service-node>.tcpip-info	show TCP/IP info
<std-super-io>.info	print information about the device
<std-super-io>.status	print status of the device
<std-telnet-console>.info	print information about the device
<std-telnet-console>.status	print status of the device
<std-telnet-console>.switch-to-host-serial-console	replace the console with a host serial console)
<std-telnet-console>.switch-to-text-console	replace the console with a text console
<std-text-console>.info	print information about the device
<std-text-console>.status	print status of the device
<std-text-console>.switch-to-host-serial-console	replace the console with a host serial console)
<std-text-console>.switch-to-telnet-console	replace the console with a telnet console
<std-text-graphics-console>.info	print information about the device
<std-text-graphics-console>.status	print status of the device
<std-text-graphics-console>.switch-to-graphics-console	replace the console with a graphics console
<std_mmc_card>.info	print information about the device
<std_mmc_card>.status	print status of the device
<std_sata_cdrom>.info	print information about the device
<std_sata_cdrom>.status	print status of the device
<std_sata_disk>.info	print information about the device
<std_sata_disk>.status	print status of the device
<SYM53C810>.info	print information about the device
<SYM53C810>.pci-header	<i>deprecated</i> — print PCI device header
<SYM53C810>.print-pci-config-reg	print PCI configuration registers
<SYM53C810>.status	print status of the device
<SYM53C875>.info	print information about the device
<SYM53C875>.pci-header	<i>deprecated</i> — print PCI device header
<SYM53C875>.print-pci-config-reg	print PCI configuration registers
<SYM53C875>.status	print status of the device
<syntable>.abi	set ABI to use
<syntable>.host-source-at	return the source code position for a given address
<syntable>.list	list source and/or disassemble
<syntable>.load-symbols	load symbols from file
<syntable>.plain-symbols	read raw symbols in nm format
<syntable>.pos	address of line or function
<syntable>.source-path	set source search path for debug info
<syntable>.whereis	find symbol by address
<sync_domain>.info	print information about the device
<sync_domain>.status	print status of the device

<system_panel_bool_in>.info	print information about the device
<system_panel_bool_in>.status	print status of the device
<system_panel_bool_out>.info	print information about the device
<system_panel_bool_out>.status	print status of the device
<system_panel_number_in>.info	print information about the device
<system_panel_number_in>.status	print status of the device
<system_panel_number_out>.info	print information about the device
<system_panel_number_out>.status	print status of the device
<system_panel_state_manager>.info	print information about the device
<system_panel_state_manager>.status	print status of the device
<tcf-agent>.info	print information about the device
<tcf-agent>.status	print status of the device
<tcf-context-proxy>.break-line	Add breakpoint at a source code line
<tcf-context-proxy>.break-location	Add breakpoint at a location
<tcf-context-proxy>.info	print information about the device
<tcf-context-proxy>.status	print status of the device
<telnet_console>.break	set a string to break on
<telnet_console>.capture-start	capture output to file
<telnet_console>.capture-stop	stop output capture to file
<telnet_console>.disconnect	terminates the telnet session
<telnet_console>.info	print information about the device
<telnet_console>.input	send string to a console
<telnet_console>.input-file	input a file into a console
<telnet_console>.kbd-abort	send a keyboard abort signal
<telnet_console>.list-break-strings	list all active string breakpoints
<telnet_console>.playback-start	start traffic generator
<telnet_console>.playback-stop	stop traffic generation
<telnet_console>.record-start	start recording of output on the console
<telnet_console>.record-stop	stop recording of output on the console
<telnet_console>.status	print status of the device
<telnet_console>.switch-to-host-serial-console	<i>deprecated</i> —
<telnet_console>.switch-to-serial-link	<i>deprecated</i> —
<telnet_console>.switch-to-text-console	<i>deprecated</i> —
<telnet_console>.unbreak	stop breaking on string
<telnet_console>.unbreak-id	remove a breakpoint
<telnet_console>.wait-for-string	wait for a string in a script branch
<telnet_console>.wait-then-write	wait for a string, then write an input string
<telnet_frontend>.info	print information about the device
<telnet_frontend>.status	print status of the device
<text-console>.break	set a string to break on
<text-console>.capture-start	capture output to file

<text-console>.capture-stop	stop output capture to file
<text-console>.close	close console window
<text-console>.disable-quiet	enable output redirection
<text-console>.disable-read-only	disable read-only mode
<text-console>.enable-quiet	disable output redirection
<text-console>.enable-read-only	enable read-only mode
<text-console>.info	print information about the device
<text-console>.input	send string to a console
<text-console>.input-file	input a file into a console
<text-console>.kbd-abort	send a keyboard abort signal
<text-console>.list-break-strings	list all active string breakpoints
<text-console>.open	open console window
<text-console>.playback-start	start traffic generator
<text-console>.playback-stop	stop traffic generation
<text-console>.quiet	<i>deprecated</i> — toggle console output redirection
<text-console>.read-only	<i>deprecated</i> — toggle read-only mode
<text-console>.record-start	start recording of output on the console
<text-console>.record-stop	stop recording of output on the console
<text-console>.status	print status of the device
<text-console>.switch-to-host-serial-console	<i>deprecated</i> —
<text-console>.switch-to-serial-link	<i>deprecated</i> —
<text-console>.switch-to-telnet-console	<i>deprecated</i> —
<text-console>.unbreak	stop breaking on string
<text-console>.unbreak-id	remove a breakpoint
<text-console>.wait-for-string	wait for a string in a script branch
<text-console>.wait-then-write	wait for a string, then write an input string
<text_panel_frontend>.info	print information about the device
<text_panel_frontend>.status	print status of the device
<top-component>.info	print information about the device
<top-component>.status	print status of the device
<trace-sync>.disable	disable core2 hardware bug workaround
<trace-sync>.enable	enable core2 hardware bug workaround
<trace-sync>.info	print information about the device
<trace-sync>.status	print status of the device
<TSB12LV26>.info	print information about the device
<TSB12LV26>.pci-header	<i>deprecated</i> — print PCI device header
<TSB12LV26>.print-pci-config-reg	print PCI configuration registers
<TSB12LV26>.status	print status of the device
<tsi500>.info	print information about the device
<tsi500>.status	print status of the device
<usb-disk>.info	print information about the device

<usb-disk>.status	print status of the device
<usb-santa>.info	print information about the device
<usb-santa>.status	print status of the device
<usb-tablet-comp>.info	print information about the device
<usb-tablet-comp>.status	print status of the device
<usb-wacom-tablet-comp>.info	print information about the device
<usb-wacom-tablet-comp>.status	print status of the device
<usb_hs_keyboard>.info	print information about the device
<usb_hs_keyboard>.key-down	print status of the device
<usb_hs_keyboard>.key-press	print information about the device
<usb_hs_keyboard>.key-up	print status of the device
<usb_hs_keyboard>.status	print information about the device
<usb_hs_keyboard_comp>.info	print status of the device
<usb_hs_keyboard_comp>.status	print information about the device
<usb_keyboard>.info	print status of the device
<usb_keyboard>.key-down	print information about the device
<usb_keyboard>.key-press	print status of the device
<usb_keyboard>.key-up	print information about the device
<usb_keyboard>.status	print status of the device
<usb_keyboard_comp>.info	print information about the device
<usb_keyboard_comp>.status	print status of the device
<usb_mouse>.info	print information about the device
<usb_mouse>.status	print status of the device
<usb_mouse_comp>.info	print information about the device
<usb_mouse_comp>.status	print status of the device
<usb_tablet>.info	print information about the device
<usb_tablet>.status	print status of the device
<usb_wacom_tablet>.info	print information about the device
<usb_wacom_tablet>.status	print status of the device
<usb_xmas_tree>.info	print information about the device
<usb_xmas_tree>.status	print status of the device
<vga_pci>.info	print information about the device
<vga_pci>.pci-header	<i>deprecated</i> — print PCI device header
<vga_pci>.print-pci-config-reg	print PCI configuration registers
<vga_pci>.redraw	Redraw display
<vga_pci>.refresh-rate	Set rate at which vga_pci updated display
<vga_pci>.status	print status of the device
<vga_pci>.text-dump	Print text contents of display
<vga_pci>.wait-for-string	Wait for substring in text mode
<vmcom>.info	print information about the device
<vmcom>.status	print status of the device

<wdb-remote>.info	print information about the device
<wdb-remote>.status	print status of the device
=	set a CLI variable
==	equal
>	greater than
>=	greater or equal
>>	bitwise right shift
@	evaluate a Python statement
[
^	bitwise XOR operation
,	evaluate a Python expression
a	alias for help-search
add-data-to-script-pipe	send data to a script pipe
add-directory	add a directory to the Simics search path
add-module-directory	add a directory to the module search path
add-pathmap-entry	add a path map entry
add-symbol-file	add symbol file to debugging contexts
alias	add an alias
and	logical and
api-apropos	alias for api-search
api-help	get API help
api-search	search API help
apropos	alias for help-search
auto-partition-configuration	suggest a cell partitioning
b	alias for break
bin	display integer in binary notation
bookmark	alias for set-bookmark
break	set breakpoint on current processor
break-cr	break on control register updates
break-exception	break on CPU exceptions
break-hap	break on haps
break-io	break on device accesses
break-line	Add breakpoint at a source code line
break-location	Add breakpoint at a location
break-log	break on log message
bt	alias for stack-trace
c	alias for run
cb	alias for cycle-break
cba	alias for cycle-break-absolute
cc	alias for run-cycles
cd	change working directory

<code>change-namespace</code>	change current namespace
<code>check-cell-partitioning</code>	verify that cell partitioning is OK
<code>clear-directories</code>	clear the Simics search path
<code>clear-memorymap</code>	clear all memory map entries
<code>clear-pathmap</code>	clear all path map entries
<code>clear-recorder</code>	clear recorded events
<code>close-tap-interface</code>	close an unused persistent TAP interface
<code>cn</code>	alias for <code>change-namespace</code>
<code>command-history</code>	show CLI command history
<code>command-list</code>	generate html document describing commands
<code>configuration-shortest-paths</code>	find shortest paths between two objects
<code>connect</code>	connect connectors
<code>connect-central</code>	connect to Simics Central
<code>connect-components</code>	<i>deprecated</i> — connect components
<code>connect-panel-to-frontend</code>	connect a simulated machine to the real network
<code>connect-real-network</code>	connect bridge between real and simulated network
<code>connect-real-network-bridge</code>	connect real host to the simulated network
<code>connect-real-network-host</code>	enable NAPT from simulated network
<code>connect-real-network-napt</code>	setup port forwarding to a simulated machine
<code>connect-real-network-port-in</code>	setup port forwarding to real machine
<code>connect-real-network-port-out</code>	<i>deprecated</i> — connect router between real and simulated network
<code>connect-real-network-router</code>	alias for <code>run</code>
<code>continue</code>	alias for <code>run-cycles</code>
<code>continue-cycles</code>	alias for <code>run-seconds</code>
<code>continue-seconds</code>	copy object
<code>copy-connector</code>	print full Simics copyright information
<code>copyright</code>	get/set default switch time
<code>cpu-switch-time</code>	create and connect memory modules to the system
<code>create-and-connect-ddr-memory</code>	create a non-instantiated cell-and-clocks
<code>create-cell-and-clocks</code>	create a non-instantiated cp3_quad100tx
<code>create-cp3-quad100tx</code>	create a non-instantiated datagram_link
<code>create-datatype-link</code>	create a non-instantiated ddr-memory-module
<code>create-ddr-memory-module</code>	create a non-instantiated ddr2-memory-module
<code>create-ddr2-memory-module</code>	create a non-instantiated ddr3-memory-module
<code>create-ddr3-memory-module</code>	create a non-instantiated dummy-component
<code>create-dummy-component</code>	create a non-instantiated etg_comp
<code>create-etg-comp</code>	create a non-instantiated etg_panel
<code>create-etg-panel</code>	create a non-instantiated eth_injector_comp
<code>create-eth-injector-comp</code>	create a non-instantiated ethernet_cable
<code>create-ethernet-cable</code>	create a non-instantiated ethernet_hub
<code>create-ethernet-hub</code>	

<code>create-ethernet-switch</code>	create a non-instantiated ethernet_switch
<code>create-ethernet-vlan-switch</code>	create a non-instantiated ethernet_vlan_switch
<code>create-example-keypad</code>	create a non-instantiated example-keypad
<code>create-example-status-panel</code>	create a non-instantiated example-status-panel
<code>create-generic-pcie-switch</code>	create a non-instantiated generic_PCIE_switch
<code>create-i2c-link-v2</code>	create a non-instantiated i2c_link_v2
<code>create-instruction-data-splitter</code>	create a non-instantiated instruction-data-splitter
<code>create-isa-fourport</code>	create a non-instantiated isa-fourport
<code>create-isa-lance</code>	create a non-instantiated isa-lance
<code>create-isa-vga</code>	create a non-instantiated isa-vga
<code>create-memory-timer</code>	create a non-instantiated memory-timer
<code>create-micron-mtfc2ggqdi-emmc-card</code>	create a non-instantiated micron_mtfc2ggqdi_emmc_card
<code>create-micron-mtfc4ggqdi-emmc-card</code>	create a non-instantiated micron_mtfc4ggqdi_emmc_card
<code>create-micron-mtfc4ggqdi-sdhc-card</code>	create a non-instantiated micron_mtfc4ggqdi_sdhc_card
<code>create-os-awareness</code>	create a non-instantiated os_awareness
<code>create-pc-dual-serial-ports</code>	create a non-instantiated pc-dual-serial-ports
<code>create-pc-floppy-controller</code>	create a non-instantiated pc-floppy-controller
<code>create-pc-quad-serial-ports</code>	create a non-instantiated pc-quad-serial-ports
<code>create-pc-single-parallel-port</code>	create a non-instantiated pc-single-parallel-port
<code>create-pci-accel-vga</code>	create a non-instantiated pci-accel-vga
<code>create-pci-am79c973</code>	create a non-instantiated pci-am79c973
<code>create-pci-bcm5703c</code>	create a non-instantiated pci-bcm5703c
<code>create-pci-bcm5704c</code>	create a non-instantiated pci-bcm5704c
<code>create-pci-dec21041</code>	create a non-instantiated pci-dec21041
<code>create-pci-dec21140a</code>	create a non-instantiated pci-dec21140a
<code>create-pci-dec21140a-dml</code>	create a non-instantiated pci-dec21140a-dml
<code>create-pci-dec21143</code>	create a non-instantiated pci-dec21143
<code>create-pci-i21152</code>	create a non-instantiated pci-i21152
<code>create-pci-i82543gc</code>	create a non-instantiated pci-i82543gc
<code>create-pci-i82546bg</code>	create a non-instantiated pci-i82546bg
<code>create-pci-i82559</code>	create a non-instantiated pci-i82559
<code>create-pci-isp1040</code>	create a non-instantiated pci-isp1040
<code>create-pci-isp2200</code>	create a non-instantiated pci-isp2200
<code>create-pci-pd6729</code>	create a non-instantiated pci-pd6729
<code>create-pci-pmc1553</code>	create a non-instantiated pci-pmc1553
<code>create-pci-sil680a</code>	create a non-instantiated pci-sil680a
<code>create-pci-sym53c810</code>	create a non-instantiated pci-sym53c810
<code>create-pci-sym53c875</code>	create a non-instantiated pci-sym53c875
<code>create-pci-sym53c876</code>	create a non-instantiated pci-sym53c876
<code>create-pci-tsb12lv26</code>	create a non-instantiated pci-tsb12lv26
<code>create-pci-vga</code>	create a non-instantiated pci-vga

<code>create-phy-comp</code>	create a a non-instantiated phy_comp
<code>create-phy-mii-transceiver</code>	create a non-instantiated phy-mii-transceiver
<code>create-ps2-keyboard-mouse</code>	create a non-instantiated ps2-keyboard-mouse
<code>create-rapidio-link</code>	create a a non-instantiated rapidio_link
<code>create-real-network-bridge</code>	create a non-instantiated real-network-bridge
<code>create-real-network-host</code>	create a non-instantiated real-network-host
<code>create-real-network-router</code>	create a non-instantiated real-network-router
<code>create-sample-gcache</code>	create a non-instantiated sample-gcache
<code>create-script-barrier</code>	create a script barrier
<code>create-script-pipe</code>	create a script pipe
<code>create-sdram-memory-module</code>	create a non-instantiated sdram-memory-module
<code>create-ser-link</code>	create a a non-instantiated ser_link
<code>create-signal-link</code>	create a a non-instantiated signal_link
<code>create-simple-fc-disk</code>	create a non-instantiated simple-fc-disk
<code>create-simple-memory-module</code>	create a a non-instantiated simple_memory_module
<code>create-sio-w83627hf</code>	create a non-instantiated sio-w83627hf
<code>create-std-etg</code>	create a non-instantiated std-etg
<code>create-std-ethernet-link</code>	create a non-instantiated std-ethernet-link
<code>create-std-firewire-bus</code>	create a non-instantiated std-firewire-bus
<code>create-std-firewire-sample-device</code>	create a non-instantiated std-firewire-sample-device
<code>create-std-generic-link</code>	create a non-instantiated std-generic-link
<code>create-std-generic-link-sample</code>	create a non-instantiated std-generic-link-sample
<code>create-std-graphics-console</code>	create a non-instantiated std-graphics-console
<code>create-std-host-serial-console</code>	create a non-instantiated std-host-serial-console
<code>create-std-ide-cdrom</code>	create a non-instantiated std-ide-cdrom
<code>create-std-ide-disk</code>	create a non-instantiated std-ide-disk
<code>create-std mmc-card</code>	create a a non-instantiated std_mmc_card
<code>create-std-ms1553-link</code>	create a non-instantiated std-ms1553-link
<code>create-std-pcmcia-flash-disk</code>	create a non-instantiated std-pcmcia-flash-disk
<code>create-std-sata-cdrom</code>	create a a non-instantiated std_sata_cdrom
<code>create-std-sata-disk</code>	create a a non-instantiated std_sata_disk
<code>create-std-scsi-bus</code>	create a non-instantiated std-scsi-bus
<code>create-std-scsi-cdrom</code>	create a non-instantiated std-scsi-cdrom
<code>create-std-scsi-disk</code>	create a non-instantiated std-scsi-disk
<code>create-std-serial-link</code>	create a non-instantiated std-serial-link
<code>create-std-server-console</code>	create a non-instantiated std-server-console
<code>create-std-service-node</code>	create a non-instantiated std-service-node
<code>create-std-super-io</code>	create a non-instantiated std-super-io
<code>create-std-telnet-console</code>	create a non-instantiated std-telnet-console
<code>create-std-text-console</code>	create a non-instantiated std-text-console
<code>create-std-text-graphics-console</code>	create a non-instantiated std-text-graphics-console

create-unconnected-ethernet-probe	
create-usb-disk	create a non-instantiated usb-disk
create-usb-hs-keyboard-comp	create a a non-instantiated usb_hs_keyboard_comp
create-usb-keyboard-comp	create a a non-instantiated usb_keyboard_comp
create-usb-mouse-comp	create a a non-instantiated usb_mouse_comp
create-usb-santa	create a non-instantiated usb-santa
create-usb-tablet-comp	create a non-instantiated usb-tablet-comp
create-usb-wacom-tablet-comp	create a non-instantiated usb-wacom-tablet-comp
create-usb-xmas-tree	create a a non-instantiated usb_xmas_tree
current-namespace	return current namespace
current-processor	return current processor
cycle-break	set cycle breakpoint
cycle-break-absolute	set absolute cycle breakpoint
da	alias for disassemble
date	host time and date
debug-context	return the current debug object
dec	display integer in decimal notation
default-port-forward-target	set default port forwarding target
defined	variable defined
delete	remove breakpoints
delete-bookmark	delete a time bookmark
devs	list all devices in Simics
digit-grouping	set output formatting for numbers
dirs	display directory stack
disable	enable/disable breakpoint
disable-hypersim	
disable-magic-breakpoint	remove breakpoint on magic break instructions
disable-mpf	disable mpf data collection
disable-multithreading	disable multithreading
disable-page-sharing	disable page-sharing
disable-real-time-mode	disable real-time behavior
disable-reverse-execution	disable reverse execution
disassemble	disassemble instructions
disassemble-settings	change disassembly output settings
disconnect	disconnect connectors
disconnect-real-network	disconnect from the real network
disconnect-real-network-port-in	remove port forwarding to a simulated machine
disconnect-real-network-port-out	remove port forwarding to real machine
display	print expression at prompt
down	go down N stack frames
dsrc-disable	disable D-SC

<code>dsti-enable</code>	enable D-STC
<code>echo</code>	echo a value to screen
<code>else</code>	
<code>enable</code>	enable/disable breakpoint
<code>enable-core2-bugfix</code>	enable hardware bug workaround
<code>enable-hypersim</code>	enables/disables hypersimulation
<code>enable-magic-breakpoint</code>	set breakpoint on magic break instructions
<code>enable-mtprof</code>	enable multithreaded simulation profiling
<code>enable-multithreading</code>	enable multithreading
<code>enable-page-sharing</code>	enable page-sharing
<code>enable-real-time-mode</code>	enable real-time behavior
<code>enable-reverse-execution</code>	enable reverse execution
<code>env</code>	
<code>etherereal</code>	alias for <code>wireshark</code>
<code>etherereal-stop</code>	alias for <code>wireshark-stop</code>
<code>except</code>	
<code>exec</code>	execute a string as a CLI command
<code>exec-info</code>	
<code>exec-info-clean</code>	
<code>exit</code>	alias for <code>quit</code>
<code>expect</code>	fail if not equal
<code>f</code>	alias for <code>frame</code>
<code>file-exists</code>	check if a file exists in the search path
<code>fin</code>	alias for <code>finish-function</code>
<code>finish</code>	alias for <code>finish-function</code>
<code>finish-function</code>	finish the current function
<code>foreach</code>	
<code>frame</code>	change current stack frame
<code>function-profile</code>	list functions sorted by profile counts
<code>get</code>	get value of physical address
<code>get-breakpoint-list</code>	return list of all breakpoints
<code>get-class-list</code>	return a list all configuration classes
<code>get-component-list</code>	return component list
<code>get-component-prefix</code>	<i>deprecated</i> — get current component name prefix
<code>get-error-command</code>	return the name of command causing error
<code>get-error-file</code>	return the file name of the CLI command error
<code>get-error-line</code>	return the file line number of the CLI command error
<code>get-error-message</code>	return the message for an error
<code>get-object-list</code>	return a list of all objects
<code>h</code>	alias for <code>help</code>
<code>help</code>	help command

help-search	search for text in documentation
hex	display integer in hexadecimal notation
hl	alias for list-haps
hypersim-status	show hypersim status
ib	alias for list-breakpoints
if	
ifm	alias for instruction-fetch-mode
ignore	set ignore count for a breakpoint
in	check for occurrence in string or list
in-list	check for occurrence in list
info-breakpoints	alias for list-breakpoints
insert-ethernet-probe	
instantiate-components	instantiate components
instruction-fetch-mode	set or get current mode for instruction fetching
int-to-double-float	interpret integer as 64-bit floating point
int-to-extended-double-float	interpret integer as 80-bit floating point
int-to-quad-float	interpret integer as 128-bit floating point
int-to-single-float	interpret integer as 32-bit floating point
interrupt-script	interrupt script execution
interrupt-script-branch	interrupt the execution of a script branch
io-stats	list most frequently accessed devices
iostc-disable	disable IO-STC
iostc-enable	enable IO-STC
istc-disable	disable I-STC
istc-enable	enable I-STC
l2p	alias for logical-to-physical
license	print Simics license
list	list source and/or disassemble
list-attributes	list all attributes
list-bookmarks	list time bookmarks
list-breakpoints	print information about breakpoints
list-checkpoints	list checkpoints
list-classes	list all configuration classes
list-components	list components
list-debug-contexts	List debug contexts
list-directories	list directories in Simics search path
list-failed-modules	list the modules that are not loadable
list-hap-callbacks	print list of hap callbacks
list-haps	print list of haps
list-hypersim-patterns	list available hypersim patterns
list-length	returns the length of a list

<code>list-modules</code>	list loadable modules
<code>list-namespaces</code>	list all namespaces
<code>list-objects</code>	list objects
<code>list-objects-with-interface</code>	Lists configuration objects implementing specific interface.
<code>list-port-forwarding-setup</code>	view the port forwarding setup
<code>list-preferences</code>	list preference
<code>list-script-branches</code>	list all script branches
<code>list-sections</code>	Lists the relocatable sections of a symbol file
<code>list-segments</code>	Lists the segments of a symbol file
<code>list-variables</code>	list CLI variables
<code>list-vars</code>	alias for <code>list-variables</code>
<code>load-binary</code>	load binary (executable) file into memory
<code>load-file</code>	load file into memory
<code>load-module</code>	load module into Simics
<code>load-persistent-state</code>	load persistent state
<code>local</code>	define a local variable
<code>log</code>	print log entries for all objects
<code>log-level</code>	set or get the global log level
<code>log-setup</code>	configure log behavior
<code>log-size</code>	set log buffer size
<code>log-type</code>	set or get the current log types
<code>logical-to-physical</code>	translate logical address to physical
<code>lookup-file</code>	alias for <code>resolve-file</code>
<code>ls</code>	list files
<code>magic-breakpoint-enabled</code>	return TRUE if the magic breakpoint is enabled
<code>man</code>	alias for <code>help</code>
<code>match-string</code>	compare string with a pattern and return matches
<code>max</code>	max
<code>min</code>	min
<code>module-list</code>	alias for <code>list-modules</code>
<code>module-list-failed</code>	alias for <code>list-failed-modules</code>
<code>module-list-refresh</code>	create a new list of loadable modules
<code>move-object</code>	move object
<code>n</code>	alias for <code>next-line</code>
<code>native-path</code>	convert a filename to host native form
<code>network-helper</code>	set/show name of host network helper
<code>new-attr-meter</code>	create a new attribute meter
<code>new-cell-and-clocks</code>	create an instantiated cell-and-clocks
<code>new-central-server</code>	create a Simics Central server
<code>new-context</code>	create a new context
<code>new-cp3-quad100tx</code>	create a an instantiated cp3_quad100tx

<code>new-datagram-link</code>	create a an instantiated datagram_link
<code>new-dummy-component</code>	create an instantiated dummy-component
<code>new-etg</code>	<i>deprecated</i> — Create an Ethernet traffic generator
<code>new-etg-comp</code>	create a an instantiated etg_comp
<code>new-etg-panel</code>	create a an instantiated etg_panel
<code>new-eth-injector-comp</code>	create a an instantiated eth_injector_comp
<code>new-ethernet-cable</code>	create a an instantiated ethernet_cable
<code>new-ethernet-hub</code>	create a an instantiated ethernet_hub
<code>new-ethernet-switch</code>	create a an instantiated ethernet_switch
<code>new-ethernet-vlan-switch</code>	create a an instantiated ethernet_vlan_switch
<code>new-example-keypad</code>	create an instantiated example-keypad
<code>new-example-status-panel</code>	create an instantiated example-status-panel
<code>new-file-cdrom</code>	create new file-cdrom object
<code>new-gdb-remote</code>	Create a gdb session
<code>new-generic-pcie-switch</code>	create a an instantiated generic_PCIE_switch
<code>new-glink</code>	create a new generic message link
<code>new-hap-meter</code>	create a new hap meter
<code>new-i2c-link-v2</code>	create a an instantiated i2c_link_v2
<code>new-mem-traffic-meter</code>	create a new memory traffic meter
<code>new-micron-mtfc2ggqdi-emmc-card</code>	create a an instantiated micron_mtfc2ggqdi_emmc_card
<code>new-micron-mtfc4ggqdi-emmc-card</code>	create a an instantiated micron_mtfc4ggqdi_emmc_card
<code>new-micron-mtfc4ggqdi-sdhc-card</code>	create a an instantiated micron_mtfc4ggqdi_sdhc_card
<code>new-os-awareness</code>	create a an instantiated os_awareness
<code>new-phy-comp</code>	create a an instantiated phy_comp
<code>new-rapidio-link</code>	create a an instantiated rapidio_link
<code>new-rapidio-tape</code>	start traffic generator
<code>new-real-network-bridge</code>	create an instantiated real-network-bridge
<code>new-real-network-host</code>	create an instantiated real-network-host
<code>new-real-network-router</code>	create an instantiated real-network-router
<code>new-realtime</code>	create a new realtime object
<code>new-remote-frontend</code>	create a remote debugger session
<code>new-sample-gcache</code>	create an instantiated sample-gcache
<code>new-ser-link</code>	create a an instantiated ser_link
<code>new-signal-link</code>	create a an instantiated signal_link
<code>new-simple-memory-module</code>	create a an instantiated simple_memory_module
<code>new-std-etg</code>	create an instantiated std-etg
<code>new-std-ethernet-link</code>	create an instantiated std-ethernet-link
<code>new-std-firewire-bus</code>	create an instantiated std-firewire-bus
<code>new-std-firewire-sample-device</code>	create an instantiated std-firewire-sample-device
<code>new-std-generic-link</code>	create an instantiated std-generic-link
<code>new-std-graphics-console</code>	create an instantiated std-graphics-console

<code>new-std-host-serial-console</code>	create an instantiated std-host-serial-console
<code>new-std-mmc-card</code>	create a an instantiated std_mmc_card
<code>new-std-ms1553-link</code>	create an instantiated std-ms1553-link
<code>new-std-pcmcia-flash-disk</code>	create an instantiated std-pcmcia-flash-disk
<code>new-std-sata-cdrom</code>	create a an instantiated std_sata_cdrom
<code>new-std-sata-disk</code>	create a an instantiated std_sata_disk
<code>new-std-scsi-bus</code>	create an instantiated std-scsi-bus
<code>new-std-serial-link</code>	create an instantiated std-serial-link
<code>new-std-server-console</code>	create an instantiated std-server-console
<code>new-std-service-node</code>	create an instantiated std-service-node
<code>new-std-telnet-console</code>	create an instantiated std-telnet-console
<code>new-std-text-console</code>	create an instantiated std-text-console
<code>new-std-text-graphics-console</code>	create an instantiated std-text-graphics-console
<code>new-symtable</code>	create new symbol table
<code>new-tcf-agent</code>	create a tcf agent
<code>new-time-server</code>	Create a new time server
<code>new-tracer</code>	create a new tracer
<code>new-usb-disk</code>	create an instantiated usb-disk
<code>new-usb-disk-from-image</code>	create new usb disk with content
<code>new-usb-hs-keyboard-comp</code>	create a an instantiated usb_hs_keyboard_comp
<code>new-usb-keyboard-comp</code>	create a an instantiated usb_keyboard_comp
<code>new-usb-mouse-comp</code>	create a an instantiated usb_mouse_comp
<code>new-usb-santa</code>	create an instantiated usb-santa
<code>new-usb-tablet-comp</code>	create an instantiated usb-tablet-comp
<code>new-usb-wacom-tablet-comp</code>	create an instantiated usb-wacom-tablet-comp
<code>new-usb-xmas-tree</code>	create a an instantiated usb_xmas_tree
<code>new-wdb-remote</code>	create a wdb session
<code>next</code>	alias for <code>next-line</code>
<code>next-instruction</code>	run to the next instruction, skipping subroutine calls
<code>next-line</code>	run to the next source line, skipping subroutine calls
<code>nexti</code>	alias for <code>next-instruction</code>
<code>ni</code>	alias for <code>next-instruction</code>
<code>not</code>	logical not
<code>object-exists</code>	check if object exists
<code>oct</code>	display integer in octal notation
<code>or</code>	logical or
<code>output-file-start</code>	send output to file
<code>output-file-stop</code>	stop sending output to file
<code>output-radix</code>	change the default output radix
<code>p</code>	alias for <code>print</code>
<code>pcap-dump</code>	Dump network traffic to a file, in libpcap format

pcap-dump-stop	stop the current dump
pcapdump	alias for pcap-dump
pcapdump-stop	alias for pcap-dump-stop
pdisable	switch processor off
penable	switch processor on
peq	alias for print-event-queue
pid	print pid of Simics process
pipe	run commands through a pipe
popd	pop directory from directory stack
pos	address of line or function
pow	power of
pregs	print cpu registers
print	display integer in various bases
print-event-queue	print event queue for processor
print-time	print number of steps and cycles executed
psel	alias for pselect
pselect	select a processor
pstatus	show processors' status
psym	print value of symbolic expression
ptime	alias for print-time
pushd	push directory on directory stack
pwd	print working directory
python	evaluate a Python expression
q	alias for quit
quit	quit from Simics
r	alias for run
range	create and return a list of integers
rc	alias for run-cycles
read-configuration	restore configuration
read-reg	read a register
read-variable	value of a named variable
readme	print information about Simics
resolve-file	resolve a filename
restart-simics	restart the current simics session
rev	alias for reverse
reverse	run simulation backwards
reverse-cycles	run simulation backwards
reverse-next-instruction	reverse to the previous instruction, skipping subroutine calls
reverse-next-line	reverse to the previous source line, skipping subroutine calls
reverse-step-instruction	reverse step one or more instruction
reverse-step-line	reverse to the previous source line

reverse-to	set a temporary time breakpoint and run backwards
revto	alias for <code>reverse-to</code>
rexec-limit	tune reverse execution performance parameters
rlimit	alias for <code>rexec-limit</code>
rn	alias for <code>reverse-next-line</code>
rnext	alias for <code>reverse-next-line</code>
rnexti	alias for <code>reverse-next-instruction</code>
rni	alias for <code>reverse-next-instruction</code>
rs	alias for <code>reverse-step-line</code>
rstep	alias for <code>reverse-step-line</code>
rstepli	alias for <code>reverse-step-instruction</code>
run	start execution
run-command-file	execute a simics script
run-cycles	start execution
run-python-file	execute Python file
run-seconds	execute for seconds
s	alias for <code>step-line</code>
save-component-template	save a component configuration template
save-persistent-state	save persistent simulator state
save-preferences	save preference
sb	alias for <code>step-break</code>
sba	alias for <code>step-break-absolute</code>
sc	alias for <code>step-cycle</code>
script-branch	check if script pipe contains data
script-pipe-has-data	alias for <code>help-search</code>
search	set profilers to display in source listing
select-profiles	set physical address to specified value
set	set a bookmark at the current point in time
set-bookmark	<i>deprecated</i> — set a prefix for all component names
set-component-prefix	set the current context of a CPU
set-context	limit memory usage
set-memory-limit	set the min latency of the sync domain
set-min-latency	set an instruction pattern for a breakpoint
set-pattern	set the current processor's program counter
set-pc	set a syntax prefix for a breakpoint
set-prefix	set a syntax substring for a breakpoint
set-substr	limit the number of simulation threads
set-thread-limit	execute a shell command
shell	show the current memory map
show-memorymap	show the current path map
show-pathmap	

si	alias for step-instruction
signed	interpret unsigned integer as signed
signed16	interpret unsigned integer as signed
signed32	interpret unsigned integer as signed
signed64	interpret unsigned integer as signed
signed8	interpret unsigned integer as signed
sim-break	alias for step-break
sim-break-absolute	alias for step-break-absolute
simulation-replaying	check if simulation is replaying
simulation-reversing	check if simulation is reversing
simulation-running	check if simulation is running
skip-to	skip to the specified point in the simulation
split-string	split string based on its type
stack-trace	display stack trace
state-assertion-connect	connect to a state-assertion receiver
state-assertion-create-file	record a state assertion file
state-assertion-open-file	open a state assertion file for comparing
state-assertion-receive	wait for a connection from a state assertion sender
state-assertion-simple-assert	assert the file
state-assertion-simple-record	record the state of an object every x steps
stc-status	show I- and D-STC status
step	alias for step-line
step-break	set time breakpoint
step-break-absolute	set absolute time breakpoint
step-cycle	step one or more cycles
step-instruction	step one or more instructions
step-line	run to the next source line
stepi	alias for step-instruction
stop	interrupt simulation
sym	alias for symval
sym-address	return the address of expression or source line
sym-file	return source file for function or address
sym-function	return function at a given address
sym-line	return source line for function or address
sym-source	print source location for function or address
sym-string	evaluate symbolic expression
sym-type	return the type of a symbolic expression
sym-value	evaluate symbolic expression
symval	evaluate symbolic expression
sync-info	Print synchronization configuration
system-info	system info

system-perfmeter	activate Simics performance monitoring
system-perfmeter-plot	create graphs from performance data
tcpdump	run the tcpdump program
tcpdump-stop	stop the current tcpdump capture
telnet-frontend	activate telnet based command line
trace-breakpoint	trace breakpoints
trace-cr	trace control register updates
trace-exception	trace exceptions
trace-hap	trace haps
trace-io	trace device accesses
trace-io-setup	
try	runs a block of commands and catches any error
turbo-disable-block-profile	
turbo-enable-block-profile	
turbo-info	
turbo-info-clean	
turbo-show-block-profile	
unbreak	remove breakpoint range
unbreak-cr	break on control register updates
unbreak-exception	break on CPU exceptions
unbreak-hap	break on haps
unbreak-io	break on device accesses
uncall	alias for uncall-function
uncall-function	reverse to when the current function was called
undisplay	remove expression installed by display
unload-module	unload module
unset	remove a CLI variable
unstep-instruction	alias for reverse-step-instruction
untrace-breakpoint	trace breakpoints
untrace-cr	trace control register updates
untrace-exception	trace exceptions
untrace-hap	trace haps
untrace-io	trace device accesses
up	go up N stack frames
version	display Simics version
wait-for-breakpoint	wait for a memory breakpoint to trigger
wait-for-hap	<i>deprecated</i> — wait until hap occurs
wait-for-script-barrier	wait until enough branches have reached a barrier
wait-for-script-pipe	wait until there is data on a script pipe
wait-for-variable	<i>deprecated</i> — wait for a variable to change
where	alias for stack-trace

whereis	find symbol by address
while	
win-about	information on GUI load failure
wireshark	run the wireshark/ethereal program
wireshark-stop	stop the current wireshark capture
write-configuration	save configuration
write-reg	write to register
x	examine raw memory contents
	bitwise OR operation
~	bitwise not

3.2 By Categories

Breakpoints

<breakpoint>.break	set breakpoint
<breakpoint>.tbreak	set temporary breakpoint on current processor
<cycle>.cycle-break	set cycle breakpoint
<cycle>.cycle-break-absolute	set absolute cycle breakpoint
break	set breakpoint on current processor
cycle-break	set cycle breakpoint
cycle-break-absolute	set absolute cycle breakpoint
delete	remove breakpoints
disable	enable/disable breakpoint
disable-magic-breakpoint	remove breakpoint on magic break instructions
enable	enable/disable breakpoint
enable-magic-breakpoint	set breakpoint on magic break instructions
get-breakpoint-list	return list of all breakpoints
ignore	set ignore count for a breakpoint
list-breakpoints	print information about breakpoints
magic-breakpoint-enabled	return TRUE if the magic breakpoint is enabled
set-pattern	set an instruction pattern for a breakpoint
set-prefix	set a syntax prefix for a breakpoint
set-substr	set a syntax substring for a breakpoint
step-break	set time breakpoint
step-break-absolute	set absolute time breakpoint
unbreak	remove breakpoint range

CD-ROM

<file-cdrom>.delete delete an unused file-cdrom object

<code>create-std-ide-cdrom</code>	create a non-instantiated std-ide-cdrom
<code>create-std-sata-cdrom</code>	create a non-instantiated std_sata_cdrom
<code>create-std-scsi-cdrom</code>	create a non-instantiated std-scsi-cdrom
<code>new-file-cdrom</code>	create new file-cdrom object
<code>new-std-sata-cdrom</code>	create a an instantiated std_sata_cdrom

Calypso

`list-objects-with-interface` Lists configuration objects implementing specific interface.

Changing Simulated State

<code><image>.set</code>	set bytes in image to specified value
<code><int_register>.write-reg</code>	write to register
<code><memory-space>.load-binary</code>	load binary (executable) file into memory
<code><memory-space>.load-file</code>	load file into memory
<code><memory-space>.set</code>	set physical address to specified value without side-effects
<code><memory-space>.write</code>	set physical address to specified value
<code><port-space>.set</code>	set physical address to specified value without side-effects
<code><port-space>.write</code>	set physical address to specified value
<code>load-binary</code>	load binary (executable) file into memory
<code>load-file</code>	load file into memory
<code>pdisable</code>	switch processor off
<code>penable</code>	switch processor on
<code>set</code>	set physical address to specified value
<code>set-pc</code>	set the current processor's program counter
<code>write-reg</code>	write to register

Command Line Interface

<code>!</code>	execute a shell command
<code>!=</code>	not equal
<code>#</code>	treat the line as a comment
<code>\$</code>	get the value of a CLI variable
<code>%</code>	read register by name, module or string formatting
<code>&</code>	bitwise AND operation
<code>*</code>	arithmetic multiplication
<code>+</code>	arithmetic addition, string and list concatenation

<code>+=</code>	set a CLI variable
<code>-</code>	arithmetic subtraction
<code>-=</code>	set a CLI variable
<code>-></code>	access object attribute
<code>/</code>	arithmetic division
<code>:</code>	access component object in a component
<code><</code>	less than
<code><<</code>	bitwise left shift
<code><=</code>	less or equal
<code><cycle>.wait-for-cycle</code>	wait until reaching cycle
<code><cycle>.wait-for-time</code>	wait until reaching a specified time
<code><int_register>.wait-for-register-read</code>	wait for a register read
<code><int_register>.wait-for-register-write</code>	wait for a register write
<code>=</code>	set a CLI variable
<code>==</code>	equal
<code>></code>	greater than
<code>>=</code>	greater or equal
<code>>></code>	bitwise right shift
<code>@</code>	evaluate a Python statement
<code>[</code>	
<code>^</code>	bitwise XOR operation
<code>,</code>	evaluate a Python expression
<code>add-data-to-script-pipe</code>	send data to a script pipe
<code>alias</code>	add an alias
<code>and</code>	logical and
<code>bin</code>	display integer in binary notation
<code>cd</code>	change working directory
<code>change-namespace</code>	change current namespace
<code>command-history</code>	show CLI command history
<code>create-script-barrier</code>	create a script barrier
<code>create-script-pipe</code>	create a script pipe
<code>current-namespace</code>	return current namespace
<code>current-processor</code>	return current processor
<code>date</code>	host time and date
<code>dec</code>	display integer in decimal notation
<code>defined</code>	variable defined
<code>digit-grouping</code>	set output formatting for numbers
<code>dirs</code>	display directory stack
<code>disassemble-settings</code>	change disassembly output settings
<code>display</code>	print expression at prompt
<code>echo</code>	echo a value to screen

else	
env	
except	
exec	execute a string as a CLI command
foreach	
get-error-command	return the name of command causing error
get-error-file	return the file name of the CLI command error
get-error-line	return the file line number of the CLI command error
get-error-message	return the message for an error
hex	display integer in hexadecimal notation
if	
in	check for occurrence in string or list
in-list	check for occurrence in list
int-to-double-float	interpret integer as 64-bit floating point
int-to-extended-double-float	interpret integer as 80-bit floating point
int-to-quad-float	interpret integer as 128-bit floating point
int-to-single-float	interpret integer as 32-bit floating point
interrupt-script	interrupt script execution
interrupt-script-branch	interrupt the execution of a script branch
list-length	returns the length of a list
list-namespaces	list all namespaces
list-script-branches	list all script branches
list-variables	list CLI variables
local	define a local variable
ls	list files
match-string	compare string with a pattern and return matches
max	max
min	min
not	logical not
object-exists	check if object exists
oct	display integer in octal notation
or	logical or
output-radix	change the default output radix
pid	print pid of Simics process
pipe	run commands through a pipe
popd	pop directory from directory stack
pow	power of
print	display integer in various bases
pselect	select a processor
pushd	push directory on directory stack
pwd	print working directory

<code>python</code>	evaluate a Python expression
<code>quit</code>	quit from Simics
<code>range</code>	create and return a list of integers
<code>read-variable</code>	value of a named variable
<code>restart-simics</code>	restart the current simics session
<code>run-command-file</code>	execute a simics script
<code>run-python-file</code>	execute Python file
<code>script-branch</code>	
<code>script-pipe-has-data</code>	check if script pipe contains data
<code>shell</code>	execute a shell command
<code>signed</code>	interpret unsigned integer as signed
<code>signed16</code>	interpret unsigned integer as signed
<code>signed32</code>	interpret unsigned integer as signed
<code>signed64</code>	interpret unsigned integer as signed
<code>signed8</code>	interpret unsigned integer as signed
<code>simulation-replaying</code>	check if simulation is replaying
<code>simulation-reversing</code>	check if simulation is reversing
<code>simulation-running</code>	check if simulation is running
<code>split-string</code>	split string based on its type
<code>try</code>	runs a block of commands and catches any error
<code>undisplay</code>	remove expression installed by display
<code>unset</code>	remove a CLI variable
<code>wait-for-breakpoint</code>	wait for a memory breakpoint to trigger
<code>wait-for-hap</code>	<i>deprecated</i> — wait until hap occurs
<code>wait-for-script-barrier</code>	wait until enough branches have reached a barrier
<code>wait-for-script-pipe</code>	wait until there is data on a script pipe
<code>wait-for-variable</code>	<i>deprecated</i> — wait for a variable to change
<code>while</code>	
<code> </code>	bitwise OR operation
<code>~</code>	bitwise not

Components

<code><component>.connect</code>	<i>deprecated</i> — connect components
<code><component>.debug-program</code>	<i>deprecated</i> — debug program
<code><component>.delete</code>	delete non-instantiated components
<code><component>.disconnect</code>	disconnect component connector
<code><component>.get-component-object</code>	get named object from components
<code><component>.get-connection</code>	return connection information
<code><component>.get-connector-list</code>	return list of connectors
<code><component>.get-processor-list</code>	return list of processors
<code>change-namespace</code>	change current namespace

<code>connect</code>	connect connectors
<code>connect-components</code>	<i>deprecated</i> — connect components
<code>copy-connector</code>	copy object
<code>create-and-connect-ddr-memory</code>	create and connect memory modules to the system
<code>create-cell-and-clocks</code>	create a non-instantiated cell-and-clocks
<code>create-cp3-quad100tx</code>	create a non-instantiated cp3_quad100tx
<code>create-datagram-link</code>	create a non-instantiated datagram_link
<code>create-ddr-memory-module</code>	create a non-instantiated ddr-memory-module
<code>create-ddr2-memory-module</code>	create a non-instantiated ddr2-memory-module
<code>create-ddr3-memory-module</code>	create a non-instantiated ddr3-memory-module
<code>create-dummy-component</code>	create a non-instantiated dummy-component
<code>create-etg-comp</code>	create a non-instantiated etg_comp
<code>create-etg-panel</code>	create a non-instantiated etg_panel
<code>create-eth-injector-comp</code>	create a non-instantiated eth_injector_comp
<code>create-ethernet-cable</code>	create a non-instantiated ethernet_cable
<code>create-ethernet-hub</code>	create a non-instantiated ethernet_hub
<code>create-ethernet-switch</code>	create a non-instantiated ethernet_switch
<code>create-ethernet-vlan-switch</code>	create a non-instantiated ethernet_vlan_switch
<code>create-example-keypad</code>	create a non-instantiated example-keypad
<code>create-example-status-panel</code>	create a non-instantiated example-status-panel
<code>create-generic-pcie-switch</code>	create a non-instantiated generic_pcie_switch
<code>create-i2c-link-v2</code>	create a non-instantiated i2c_link_v2
<code>create-instruction-data-splitter</code>	create a non-instantiated instruction-data-splitter
<code>create-isa-fourport</code>	create a non-instantiated isa-fourport
<code>create-isa-lance</code>	create a non-instantiated isa-lance
<code>create-isa-vga</code>	create a non-instantiated isa-vga
<code>create-memory-timer</code>	create a non-instantiated memory-timer
<code>create-micron-mtfc2ggqdi-emmc-card</code>	create a non-instantiated micron_mtfc2ggqdi_emmc_card
<code>create-micron-mtfc4ggqdi-emmc-card</code>	create a non-instantiated micron_mtfc4ggqdi_emmc_card
<code>create-micron-mtfc4ggqdi-sdhc-card</code>	create a non-instantiated micron_mtfc4ggqdi_sdhc_card
<code>create-os-awareness</code>	create a non-instantiated os Awareness
<code>create-pc-dual-serial-ports</code>	create a non-instantiated pc-dual-serial-ports
<code>create-pc-floppy-controller</code>	create a non-instantiated pc-floppy-controller
<code>create-pc-quad-serial-ports</code>	create a non-instantiated pc-quad-serial-ports
<code>create-pc-single-parallel-port</code>	create a non-instantiated pc-single-parallel-port
<code>create-pci-accel-vga</code>	create a non-instantiated pci-accel-vga
<code>create-pci-am79c973</code>	create a non-instantiated pci-am79c973
<code>create-pci-bcm5703c</code>	create a non-instantiated pci-bcm5703c
<code>create-pci-bcm5704c</code>	create a non-instantiated pci-bcm5704c
<code>create-pci-dec21041</code>	create a non-instantiated pci-dec21041
<code>create-pci-dec21140a</code>	create a non-instantiated pci-dec21140a

create-pci-dec21140a-dml	create a non-instantiated pci-dec21140a-dml
create-pci-dec21143	create a non-instantiated pci-dec21143
create-pci-i21152	create a non-instantiated pci-i21152
create-pci-i82543gc	create a non-instantiated pci-i82543gc
create-pci-i82546bg	create a non-instantiated pci-i82546bg
create-pci-i82559	create a non-instantiated pci-i82559
create-pci-isp1040	create a non-instantiated pci-isp1040
create-pci-isp2200	create a non-instantiated pci-isp2200
create-pci-pd6729	create a non-instantiated pci-pd6729
create-pci-pmc1553	create a non-instantiated pci-pmc1553
create-pci-sil680a	create a non-instantiated pci-sil680a
create-pci-sym53c810	create a non-instantiated pci-sym53c810
create-pci-sym53c875	create a non-instantiated pci-sym53c875
create-pci-sym53c876	create a non-instantiated pci-sym53c876
create-pci-tsb12lv26	create a non-instantiated pci-tsb12lv26
create-pci-vga	create a non-instantiated pci-vga
create-phy-comp	create a non-instantiated phy_comp
create-phy-mii-transceiver	create a non-instantiated phy-mii-transceiver
create-ps2-keyboard-mouse	create a non-instantiated ps2-keyboard-mouse
create-rapidio-link	create a non-instantiated rapidio_link
create-real-network-bridge	create a non-instantiated real-network-bridge
create-real-network-host	create a non-instantiated real-network-host
create-real-network-router	create a non-instantiated real-network-router
create-sample-gcache	create a non-instantiated sample-gcache
create-sdram-memory-module	create a non-instantiated sdram-memory-module
create-ser-link	create a non-instantiated ser_link
create-signal-link	create a non-instantiated signal_link
create-simple-fc-disk	create a non-instantiated simple-fc-disk
create-simple-memory-module	create a non-instantiated simple_memory_module
create-sio-w83627hf	create a non-instantiated sio-w83627hf
create-std-etg	create a non-instantiated std-etg
create-std-ethernet-link	create a non-instantiated std-ethernet-link
create-std-firewire-bus	create a non-instantiated std-firewire-bus
create-std-firewire-sample-device	create a non-instantiated std-firewire-sample-device
create-std-generic-link	create a non-instantiated std-generic-link
create-std-generic-link-sample	create a non-instantiated std-generic-link-sample
create-std-graphics-console	create a non-instantiated std-graphics-console
create-std-host-serial-console	create a non-instantiated std-host-serial-console
create-std-ide-cdrom	create a non-instantiated std-ide-cdrom
create-std-ide-disk	create a non-instantiated std-ide-disk
create-std mmc-card	create a non-instantiated std_mmc_card

<code>create-std-ms1553-link</code>	create a non-instantiated std-ms1553-link
<code>create-std-pcmcia-flash-disk</code>	create a non-instantiated std-pcmcia-flash-disk
<code>create-std-sata-cdrom</code>	create a a non-instantiated std_sata_cdrom
<code>create-std-sata-disk</code>	create a a non-instantiated std_sata_disk
<code>create-std-scsi-bus</code>	create a non-instantiated std-scsi-bus
<code>create-std-scsi-cdrom</code>	create a non-instantiated std-scsi-cdrom
<code>create-std-scsi-disk</code>	create a non-instantiated std-scsi-disk
<code>create-std-serial-link</code>	create a non-instantiated std-serial-link
<code>create-std-server-console</code>	create a non-instantiated std-server-console
<code>create-std-service-node</code>	create a non-instantiated std-service-node
<code>create-std-super-io</code>	create a non-instantiated std-super-io
<code>create-std-telnet-console</code>	create a non-instantiated std-telnet-console
<code>create-std-text-console</code>	create a non-instantiated std-text-console
<code>create-std-text-graphics-console</code>	create a non-instantiated std-text-graphics-console
<code>create-usb-disk</code>	create a non-instantiated usb-disk
<code>create-usb-hs-keyboard-comp</code>	create a a non-instantiated usb_hs_keyboard_comp
<code>create-usb-keyboard-comp</code>	create a a non-instantiated usb_keyboard_comp
<code>create-usb-mouse-comp</code>	create a a non-instantiated usb_mouse_comp
<code>create-usb-santa</code>	create a non-instantiated usb-santa
<code>create-usb-tablet-comp</code>	create a non-instantiated usb-tablet-comp
<code>create-usb-wacom-tablet-comp</code>	create a non-instantiated usb-wacom-tablet-comp
<code>create-usb-xmas-tree</code>	create a a non-instantiated usb_xmas_tree
<code>current-namespace</code>	return current namespace
<code>disconnect</code>	disconnect connectors
<code>get-component-list</code>	return component list
<code>get-component-prefix</code>	<i>deprecated</i> — get current component name prefix
<code>instantiate-components</code>	instantiate components
<code>list-components</code>	list components
<code>move-object</code>	move object
<code>new-cell-and-clocks</code>	create an instantiated cell-and-clocks
<code>new-cp3-quad100tx</code>	create a an instantiated cp3_quad100tx
<code>new-datagram-link</code>	create a an instantiated datagram_link
<code>new-dummy-component</code>	create an instantiated dummy-component
<code>new-etg-comp</code>	create a an instantiated etg_comp
<code>new-etg-panel</code>	create a an instantiated etg_panel
<code>new-eth-injector-comp</code>	create a an instantiated eth_injector_comp
<code>new-ethernet-cable</code>	create a an instantiated ethernet_cable
<code>new-ethernet-hub</code>	create a an instantiated ethernet_hub
<code>new-ethernet-switch</code>	create a an instantiated ethernet_switch
<code>new-ethernet-vlan-switch</code>	create a an instantiated ethernet_vlan_switch
<code>new-example-keypad</code>	create an instantiated example-keypad

3.2. By Categories

new-example-status-panel	create an instantiated example-status-panel
new-generic-pcie-switch	create a an instantiated generic_pcie_switch
new-i2c-link-v2	create a an instantiated i2c_link_v2
new-micron-mtfc2ggqdi-emmc-card	create a an instantiated micron_mtfc2ggqdi_emmc_card
new-micron-mtfc4ggqdi-emmc-card	create a an instantiated micron_mtfc4ggqdi_emmc_card
new-micron-mtfc4ggqdi-sdhc-card	create a an instantiated micron_mtfc4ggqdi_sdhc_card
new-os-awareness	create a an instantiated os_awareness
new-phy-comp	create a an instantiated phy_comp
new-rapidio-link	create a an instantiated rapidio_link
new-real-network-bridge	create an instantiated real-network-bridge
new-real-network-host	create an instantiated real-network-host
new-real-network-router	create an instantiated real-network-router
new-sample-gcache	create an instantiated sample-gcache
new-ser-link	create a an instantiated ser_link
new-signal-link	create a an instantiated signal_link
new-simple-memory-module	create a an instantiated simple_memory_module
new-std-etg	create an instantiated std-etg
new-std-ethernet-link	create an instantiated std-ethernet-link
new-std-firewire-bus	create an instantiated std-firewire-bus
new-std-firewire-sample-device	create an instantiated std-firewire-sample-device
new-std-generic-link	create an instantiated std-generic-link
new-std-graphics-console	create an instantiated std-graphics-console
new-std-host-serial-console	create an instantiated std-host-serial-console
new-std mmc-card	create a an instantiated std_mmc_card
new-std-ms1553-link	create an instantiated std-ms1553-link
new-std-pcmcia-flash-disk	create an instantiated std-pcmcia-flash-disk
new-std-sata-cdrom	create a an instantiated std_sata_cdrom
new-std-sata-disk	create a an instantiated std_sata_disk
new-std-scsi-bus	create an instantiated std-scsi-bus
new-std-serial-link	create an instantiated std-serial-link
new-std-server-console	create an instantiated std-server-console
new-std-service-node	create an instantiated std-service-node
new-std-telnet-console	create an instantiated std-telnet-console
new-std-text-console	create an instantiated std-text-console
new-std-text-graphics-console	create an instantiated std-text-graphics-console
new-usb-disk	create an instantiated usb-disk
new-usb-hs-keyboard-comp	create a an instantiated usb_hs_keyboard_comp
new-usb-keyboard-comp	create a an instantiated usb_keyboard_comp
new-usb-mouse-comp	create a an instantiated usb_mouse_comp
new-usb-santa	create an instantiated usb-santa
new-usb-tablet-comp	create an instantiated usb-tablet-comp

<code>new-usb-wacom-tablet-comp</code>	create an instantiated usb-wacom-tablet-comp
<code>new-usb-xmas-tree</code>	create a an instantiated usb_xmas_tree
<code>set-component-prefix</code>	<i>deprecated</i> — set a prefix for all component names

Configuration

<code><image>.add-diff-file</code>	add a diff file to the image
<code><image>.add-partial-diff-file</code>	add a partial diff file to the image
<code><image>.save</code>	save image to disk
<code><image>.save-diff-file</code>	save changes since last checkpoint
<code><memory-space>.add-map</code>	map device in a memory-space
<code><memory-space>.del-map</code>	remove device map from a memory-space
<code><memory-space>.map</code>	list memory map
<code><port-space>.add-map</code>	map device in a port-space
<code><port-space>.del-map</code>	remove device map from a port-space
<code><port-space>.map</code>	list port map
<code>add-directory</code>	add a directory to the Simics search path
<code>add-module-directory</code>	add a directory to the module search path
<code>auto-partition-configuration</code>	suggest a cell partitioning
<code>check-cell-partitioning</code>	verify that cell partitioning is OK
<code>configuration-shortest-paths</code>	find shortest paths between two objects
<code>devs</code>	list all devices in Simics
<code>get-class-list</code>	return a list all configuration classes
<code>get-object-list</code>	return a list of all objects
<code>list-attributes</code>	list all attributes
<code>list-checkpoints</code>	list checkpoints
<code>list-classes</code>	list all configuration classes
<code>list-failed-modules</code>	list the modules that are not loadable
<code>list-modules</code>	list loadable modules
<code>list-objects</code>	list objects
<code>list-preferences</code>	list preference
<code>load-module</code>	load module into Simics
<code>load-persistent-state</code>	load persistent state
<code>module-list-refresh</code>	create a new list of loadable modules
<code>read-configuration</code>	restore configuration
<code>save-component-template</code>	save a component configuration template
<code>save-persistent-state</code>	save persistent simulator state
<code>save-preferences</code>	save preference
<code>set-min-latency</code>	set the min latency of the sync domain
<code>sync-info</code>	Print synchronization configuration

`write-configuration` save configuration

Debugging

<code><breakpoint>.break</code>	set breakpoint
<code><breakpoint>.tbreak</code>	set temporary breakpoint on current processor
<code><context>.finish-function</code>	finish the current function
<code><context>.next-instruction</code>	run to the next instruction, skipping subroutine calls
<code><context>.next-line</code>	run to the next source line, skipping subroutine calls
<code><context>.off</code>	switch off context object
<code><context>.on</code>	switch on context object
<code><context>.reverse-next-instruction</code>	reverse to the previous instruction, skipping subroutine calls
<code><context>.reverse-next-line</code>	reverse to the previous source line, skipping subroutine calls
<code><context>.reverse-step-instruction</code>	reverse to the previous instruction
<code><context>.reverse-step-line</code>	reverse to the previous source line
<code><context>.step-instruction</code>	run to the next instruction
<code><context>.step-line</code>	run to the next source line
<code><context>.uncall-function</code>	reverse to when the current function was called
<code><cycle>.cycle-break</code>	set cycle breakpoint
<code><cycle>.cycle-break-absolute</code>	set absolute cycle breakpoint
<code><memory_space>.debug</code>	get debug object
<code><os_awareness>.active-node</code>	list the active nodes
<code><os_awareness>.break-enter</code>	break the simulation when a processor becomes active
<code><os_awareness>.break-exit</code>	break the simulation when a processor becomes active
<code><os_awareness>.find</code>	find a node
<code><os_awareness>.linux-autodetect-settings</code>	Try to autodetect settings for the Linux tracker
<code><os_awareness>.list</code>	list processes/tasks
<code><os_awareness>.load-parameters</code>	Load tracker parameter settings
<code><os_awareness>.log-syscalls</code>	start logging system calls
<code><os_awareness>.node-info</code>	show software tracker node information
<code><os_awareness>.node-tree</code>	list software node tree
<code><os_awareness>.ose-detect-settings</code>	detect settings for the OSE tracker
<code><os_awareness>.partition-settings</code>	configure the partition tracker
<code><os_awareness>.qnx-detect-settings</code>	detect settings for the QNX tracker
<code><os_awareness>.track</code>	let a context track a process, thread, task, etc.
<code><os_awareness>.unbreak</code>	cancel a software breakpoint
<code><os_awareness>.vxworks-detect-settings</code>	detect settings for the VxWorks tracker
<code><os_awareness>.wr-hypervisor-detect-settings</code>	detect settings for the Wind River hypervisor tracker
<code><symtable>.list</code>	list source and/or disassemble
<code><symtable>.pos</code>	address of line or function

<symtable>.whereis	find symbol by address
add-pathmap-entry	add a path map entry
add-symbol-file	add symbol file to debugging contexts
break	set breakpoint on current processor
clear-memorymap	clear all memory map entries
clear-pathmap	clear all path map entries
cycle-break	set cycle breakpoint
cycle-break-absolute	set absolute cycle breakpoint
debug-context	return the current debug object
delete	remove breakpoints
disable	enable/disable breakpoint
disable-magic-breakpoint	remove breakpoint on magic break instructions
down	go down N stack frames
enable	enable/disable breakpoint
enable-magic-breakpoint	set breakpoint on magic break instructions
finish-function	finish the current function
frame	change current stack frame
get-breakpoint-list	return list of all breakpoints
ignore	set ignore count for a breakpoint
list	list source and/or disassemble
list-breakpoints	print information about breakpoints
list-debug-contexts	List debug contexts
list-sections	Lists the relocatable sections of a symbol file
list-segments	Lists the segments of a symbol file
magic-breakpoint-enabled	return TRUE if the magic breakpoint is enabled
new-context	create a new context
new-gdb-remote	Create a gdb session
new-remote-frontend	create a remote debugger session
new-tcf-agent	create a tcf agent
new-wdb-remote	create a wdb session
next-instruction	run to the next instruction, skipping subroutine calls
next-line	run to the next source line, skipping subroutine calls
pos	address of line or function
psym	print value of symbolic expression
reverse-next-instruction	reverse to the previous instruction, skipping subroutine calls
reverse-next-line	reverse to the previous source line, skipping subroutine calls
reverse-step-line	reverse to the previous source line
set-context	set the current context of a CPU
set-pattern	set an instruction pattern for a breakpoint
set-prefix	set a syntax prefix for a breakpoint
set-substr	set a syntax substring for a breakpoint

<code>show-memorymap</code>	show the current memory map
<code>show-pathmap</code>	show the current path map
<code>stack-trace</code>	display stack trace
<code>step-break</code>	set time breakpoint
<code>step-break-absolute</code>	set absolute time breakpoint
<code>step-line</code>	run to the next source line
<code>sym-address</code>	return the address of expression or source line
<code>sym-file</code>	return source file for function or address
<code>sym-function</code>	return function at a given address
<code>sym-line</code>	return source line for function or address
<code>sym-source</code>	print source location for function or address
<code>sym-string</code>	evaluate symbolic expression
<code>sym-type</code>	return the type of a symbolic expression
<code>sym-value</code>	evaluate symbolic expression
<code>symval</code>	evaluate symbolic expression
<code>unbreak</code>	remove breakpoint range
<code>uncall-function</code>	reverse to when the current function was called
<code>up</code>	go up N stack frames
<code>whereis</code>	find symbol by address

Disk

<code><image>.add-diff-file</code>	add a diff file to the image
<code><image>.add-partial-diff-file</code>	add a partial diff file to the image
<code><image>.save</code>	save image to disk
<code><image>.save-diff-file</code>	save changes since last checkpoint
<code><image>.set</code>	set bytes in image to specified value
<code>load-persistent-state</code>	load persistent state
<code>new-file-cdrom</code>	create new file-cdrom object
<code>new-usb-disk-from-image</code>	create new usb disk with content
<code>save-persistent-state</code>	save persistent simulator state

Ethernet

<code><BCM5703C>.connect</code>	<i>deprecated</i> — connect to a simulated Ethernet link
<code><BCM5703C>.disconnect</code>	<i>deprecated</i> — disconnect from simulated link
<code><BCM5704C>.connect</code>	<i>deprecated</i> — connect to a simulated Ethernet link
<code><BCM5704C>.disconnect</code>	<i>deprecated</i> — disconnect from simulated link
<code><DEC21041>.connect</code>	<i>deprecated</i> — connect to a simulated Ethernet link
<code><DEC21041>.disconnect</code>	<i>deprecated</i> — disconnect from simulated link
<code><DEC21140A>.connect</code>	<i>deprecated</i> — connect to a simulated Ethernet link

<DEC21140A>.disconnect	<i>deprecated</i> — disconnect from simulated link
<DEC21143>.connect	<i>deprecated</i> — connect to a simulated Ethernet link
<DEC21143>.disconnect	<i>deprecated</i> — disconnect from simulated link
<eth-transceiver>.connect	<i>deprecated</i> — connect to a simulated Ethernet link
<eth-transceiver>.disconnect	<i>deprecated</i> — disconnect from simulated link
<i2s2543>.connect	<i>deprecated</i> — connect to a simulated Ethernet link
<i2s2543>.disconnect	<i>deprecated</i> — disconnect from simulated link
<i2s2546>.connect	<i>deprecated</i> — connect to a simulated Ethernet link
<i2s2546>.disconnect	<i>deprecated</i> — disconnect from simulated link
<mii-transceiver>.connect	<i>deprecated</i> — connect to a simulated Ethernet link
<mii-transceiver>.disconnect	<i>deprecated</i> — disconnect from simulated link
create-cp3-quad100tx	create a a non-instantiated cp3_quad100tx
create-etg-comp	create a a non-instantiated etg_comp
create-eth-injector-comp	create a a non-instantiated eth_injector_comp
create-ethernet-cable	create a a non-instantiated ethernet_cable
create-ethernet-switch	create a a non-instantiated ethernet_switch
create-ethernet-vlan-switch	create a a non-instantiated ethernet_vlan_switch
create-pci-am79c973	create a non-instantiated pci-am79c973
create-pci-bcm5703c	create a non-instantiated pci-bcm5703c
create-pci-bcm5704c	create a non-instantiated pci-bcm5704c
create-pci-dec21041	create a non-instantiated pci-dec21041
create-pci-dec21140a	create a non-instantiated pci-dec21140a
create-pci-dec21140a-dml	create a non-instantiated pci-dec21140a-dml
create-pci-dec21143	create a non-instantiated pci-dec21143
create-pci-i2s2543gc	create a non-instantiated pci-i2s2543gc
create-pci-i2s2546bg	create a non-instantiated pci-i2s2546bg
create-pci-i2s2559	create a non-instantiated pci-i2s2559
create-phy-comp	create a a non-instantiated phy_comp
create-std-etg	create a non-instantiated std-etg
create-std-ethernet-link	create a non-instantiated std-ethernet-link
new-cp3-quad100tx	create a an instantiated cp3_quad100tx
new-etg	<i>deprecated</i> — Create an Ethernet traffic generator
new-etg-comp	create a an instantiated etg_comp
new-eth-injector-comp	create a an instantiated eth_injector_comp
new-ethernet-cable	create a an instantiated ethernet_cable
new-ethernet-switch	create a an instantiated ethernet_switch
new-ethernet-vlan-switch	create a an instantiated ethernet_vlan_switch
new-phy-comp	create a an instantiated phy_comp
new-std-etg	create an instantiated std-etg
new-std-ethernet-link	create an instantiated std-ethernet-link

Execution

<cell>.cpu-switch-time	get/set the time quantum for a given cell
<cycle>.cycle-break	set cycle breakpoint
<cycle>.cycle-break-absolute	set absolute cycle breakpoint
<cycle>.print-time	print number of steps and cycles executed
<memory-space>.load-binary	load binary (executable) file into memory
<realtime>.disable	disable real-time behavior
<realtime>.enable	enable real-time behavior
cpu-switch-time	get/set default switch time
cycle-break	set cycle breakpoint
cycle-break-absolute	set absolute cycle breakpoint
disable-multithreading	disable multithreading
disable-page-sharing	disable page-sharing
disable-real-time-mode	disable real-time behavior
disassemble	disassemble instructions
disassemble-settings	change disassembly output settings
dstc-disable	disable D-STC
dstc-enable	enable D-STC
enable-multithreading	enable multithreading
enable-page-sharing	enable page-sharing
enable-real-time-mode	enable real-time behavior
instruction-fetch-mode	set or get current mode for instruction fetching
iostc-disable	disable IO-STC
iostc-enable	enable IO-STC
istc-disable	disable I-STC
istc-enable	enable I-STC
load-binary	load binary (executable) file into memory
new-realtime	create a new realtime object
pdisable	switch processor off
penable	switch processor on
print-time	print number of steps and cycles executed
pstatus	show processors' status
run	start execution
run-cycles	start execution
run-seconds	execute for seconds
set-pc	set the current processor's program counter
set-thread-limit	limit the number of simulation threads
stc-status	show I- and D-STC status
step-break	set time breakpoint
step-break-absolute	set absolute time breakpoint
step-cycle	step one or more cycles
step-instruction	step one or more instructions

`stop` interrupt simulation

Fibre Channel

<code>create-pci-isp2200</code>	create a non-instantiated pci-isp2200
<code>create-simple-fc-disk</code>	create a non-instantiated simple-fc-disk

Files and Directories

<code>add-directory</code>	add a directory to the Simics search path
<code>add-module-directory</code>	add a directory to the module search path
<code>cd</code>	change working directory
<code>clear-directories</code>	clear the Simics search path
<code>dirs</code>	display directory stack
<code>file-exists</code>	check if a file exists in the search path
<code>list-checkpoints</code>	list checkpoints
<code>list-directories</code>	list directories in Simics search path
<code>list-preferences</code>	list preference
<code>ls</code>	list files
<code>native-path</code>	convert a filename to host native form
<code>output-file-start</code>	send output to file
<code>output-file-stop</code>	stop sending output to file
<code>popd</code>	pop directory from directory stack
<code>pushd</code>	push directory on directory stack
<code>pwd</code>	print working directory
<code>resolve-file</code>	resolve a filename
<code>run-command-file</code>	execute a simics script
<code>run-python-file</code>	execute Python file
<code>save-preferences</code>	save preference

GUI

`win-about` information on GUI load failure

Generic Link

<code>create-std-generic-link</code>	create a non-instantiated std-generic-link
<code>create-std-generic-link-sample</code>	create a non-instantiated std-generic-link-sample

[new-std-generic-link](#) create an instantiated std-generic-link

Graphics Adapter

[create-pci-accel-vga](#) create a non-instantiated pci-accel-vga
[create-pci-vga](#) create a non-instantiated pci-vga

Graphics Console

[create-std-graphics-console](#) create a non-instantiated std-graphics-console
[create-std-text-graphics-console](#) create a non-instantiated std-text-graphics-console
[new-std-graphics-console](#) create an instantiated std-graphics-console
[new-std-text-graphics-console](#) create an instantiated std-text-graphics-console

Haps

[list-hap-callbacks](#) print list of hap callbacks
[list-haps](#) print list of haps

Help

[api-help](#) get API help
[api-search](#) search API help
[command-list](#) generate html document describing commands
[copyright](#) print full Simics copyright information
[help](#) help command
[help-search](#) search for text in documentation
[license](#) print Simics license
[list-attributes](#) list all attributes
[list-namespaces](#) list all namespaces
[readme](#) print information about Simics
[version](#) display Simics version

IDE

[create-pci-sil680a](#) create a non-instantiated pci-sil680a

<code>create-std-ide-cdrom</code>	create a non-instantiated std-ide-cdrom
<code>create-std-ide-disk</code>	create a non-instantiated std-ide-disk

Inspecting Simulated State

<code>%</code>	read register by name, module or string formatting
<code><image>.x</code>	examine image data
<code><int_register>.read-reg</code>	read a register
<code><int_register>.register-number</code>	<i>deprecated</i> — get the number of a processor register
<code><memory-space>.get</code>	get value of physical address without side-effects
<code><memory-space>.map</code>	list memory map
<code><memory-space>.read</code>	get value of physical address
<code><memory-space>.x</code>	examine raw memory contents
<code><port-space>.get</code>	get value of physical address without side-effects
<code><port-space>.map</code>	list port map
<code><port-space>.read</code>	get value of physical address
<code>devs</code>	list all devices in Simics
<code>disassemble</code>	disassemble instructions
<code>get</code>	get value of physical address
<code>logical-to-physical</code>	translate logical address to physical
<code>pregs</code>	print cpu registers
<code>print-event-queue</code>	print event queue for processor
<code>pstatus</code>	show processors' status
<code>read-reg</code>	read a register
<code>x</code>	examine raw memory contents

Keypad

<code>create-example-keypad</code>	create a non-instantiated example-keypad
<code>new-example-keypad</code>	create an instantiated example-keypad

Logging

<code>log</code>	print log entries for all objects
<code>log-level</code>	set or get the global log level
<code>log-setup</code>	configure log behavior
<code>log-size</code>	set log buffer size
<code>log-type</code>	set or get the current log types

MMC

<code>create-micron-mtfc2ggqdi-emmc-card</code>	create a non-instantiated <code>micron_mtfc2ggqdi_emmc_card</code>
<code>create-micron-mtfc4ggqdi-emmc-card</code>	create a non-instantiated <code>micron_mtfc4ggqdi_emmc_card</code>
<code>create-micron-mtfc4ggqdi-sdhc-card</code>	create a non-instantiated <code>micron_mtfc4ggqdi_sdhc_card</code>
<code>create-std mmc-card</code>	create a non-instantiated <code>std_mmc_card</code>
<code>new-micron-mtfc2ggqdi-emmc-card</code>	create an instantiated <code>micron_mtfc2ggqdi_emmc_card</code>
<code>new-micron-mtfc4ggqdi-emmc-card</code>	create an instantiated <code>micron_mtfc4ggqdi_emmc_card</code>
<code>new-micron-mtfc4ggqdi-sdhc-card</code>	create an instantiated <code>micron_mtfc4ggqdi_sdhc_card</code>
<code>new-std mmc-card</code>	create an instantiated <code>std_mmc_card</code>

Memory

<code><image>.add-diff-file</code>	add a diff file to the image
<code><image>.add-partial-diff-file</code>	add a partial diff file to the image
<code><image>.save</code>	save image to disk
<code><image>.save-diff-file</code>	save changes since last checkpoint
<code><image>.set</code>	set bytes in image to specified value
<code><image>.x</code>	examine image data
<code><memory-space>.add-map</code>	map device in a memory-space
<code><memory-space>.del-map</code>	remove device map from a memory-space
<code><memory-space>.get</code>	get value of physical address without side-effects
<code><memory-space>.load-binary</code>	load binary (executable) file into memory
<code><memory-space>.load-file</code>	load file into memory
<code><memory-space>.map</code>	list memory map
<code><memory-space>.read</code>	get value of physical address
<code><memory-space>.set</code>	set physical address to specified value without side-effects
<code><memory-space>.write</code>	set physical address to specified value
<code><memory-space>.x</code>	examine raw memory contents
<code><port-space>.add-map</code>	map device in a port-space
<code><port-space>.del-map</code>	remove device map from a port-space
<code><port-space>.get</code>	get value of physical address without side-effects
<code><port-space>.map</code>	list port map
<code><port-space>.read</code>	get value of physical address
<code><port-space>.set</code>	set physical address to specified value without side-effects
<code><port-space>.write</code>	set physical address to specified value
<code>disassemble</code>	disassemble instructions
<code>disassemble-settings</code>	change disassembly output settings
<code>get</code>	get value of physical address
<code>load-binary</code>	load binary (executable) file into memory
<code>load-file</code>	load file into memory
<code>logical-to-physical</code>	translate logical address to physical

set set physical address to specified value
x examine raw memory contents

Modules

add-module-directory	add a directory to the module search path
list-failed-modules	list the modules that are not loadable
list-modules	list loadable modules
load-module	load module into Simics
module-list-refresh	create a new list of loadable modules

Networking

<code><BCM5703C>.connect</code>	<i>deprecated</i> — connect to a simulated Ethernet link
<code><BCM5703C>.disconnect</code>	<i>deprecated</i> — disconnect from simulated link
<code><BCM5704C>.connect</code>	<i>deprecated</i> — connect to a simulated Ethernet link
<code><BCM5704C>.disconnect</code>	<i>deprecated</i> — disconnect from simulated link
<code><DEC21041>.connect</code>	<i>deprecated</i> — connect to a simulated Ethernet link
<code><DEC21041>.disconnect</code>	<i>deprecated</i> — disconnect from simulated link
<code><DEC21140A>.connect</code>	<i>deprecated</i> — connect to a simulated Ethernet link
<code><DEC21140A>.disconnect</code>	<i>deprecated</i> — disconnect from simulated link
<code><DEC21143>.connect</code>	<i>deprecated</i> — connect to a simulated Ethernet link
<code><DEC21143>.disconnect</code>	<i>deprecated</i> — disconnect from simulated link
<code><eth-transceiver>.connect</code>	<i>deprecated</i> — connect to a simulated Ethernet link
<code><eth-transceiver>.disconnect</code>	<i>deprecated</i> — disconnect from simulated link
<code><ethernet-link>.connect-real-network-bridge</code>	connect to the real network
<code><ethernet-link>.connect-real-network-host</code>	connect to the real network
<code><ethernet-link>.connect-real-network-napt</code>	enable NAPT from simulated network
<code><ethernet-link>.disconnect-real-network</code>	disconnect from the real network
<code><ethernet_cable>.connect-real-network-bridge</code>	connect to the real network
<code><ethernet_cable>.connect-real-network-host</code>	connect to the real network
<code><ethernet_cable>.connect-real-network-napt</code>	enable NAPT from simulated network
<code><ethernet_cable>.disconnect-real-network</code>	disconnect from the real network
<code><ethernet_hub>.connect-real-network-bridge</code>	connect to the real network
<code><ethernet_hub>.connect-real-network-host</code>	connect to the real network
<code><ethernet_hub>.connect-real-network-napt</code>	enable NAPT from simulated network
<code><ethernet_hub>.disconnect-real-network</code>	disconnect from the real network
<code><ethernet_switch>.connect-real-network-bridge</code>	connect to the real network
<code><ethernet_switch>.connect-real-network-host</code>	connect to the real network

<ethernet_switch>.connect-real-network-napt	enable NAPT from simulated network
<ethernet_switch>.disconnect-real-network	disconnect from the real network
<ethernet_vlan_switch>.connect-real-network-napt	enable NAPT from simulated network
<i82543>.connect	<i>deprecated</i> — connect to a simulated Ethernet link
<i82543>.disconnect	<i>deprecated</i> — disconnect from simulated link
<i82546>.connect	<i>deprecated</i> — connect to a simulated Ethernet link
<i82546>.disconnect	<i>deprecated</i> — disconnect from simulated link
<mii-transceiver>.connect	<i>deprecated</i> — connect to a simulated Ethernet link
<mii-transceiver>.disconnect	<i>deprecated</i> — disconnect from simulated link
connect-real-network-bridge	connect bridge between real and simulated network
connect-real-network-host	connect real host to the simulated network
disconnect-real-network	disconnect from the real network
network-helper	set/show name of host network helper

Output

bin	display integer in binary notation
dec	display integer in decimal notation
digit-grouping	set output formatting for numbers
disassemble-settings	change disassembly output settings
display	print expression at prompt
echo	echo a value to screen
hex	display integer in hexadecimal notation
int-to-double-float	interpret integer as 64-bit floating point
int-to-extended-double-float	interpret integer as 80-bit floating point
int-to-quad-float	interpret integer as 128-bit floating point
int-to-single-float	interpret integer as 32-bit floating point
log	print log entries for all objects
log-level	set or get the global log level
log-setup	configure log behavior
log-size	set log buffer size
log-type	set or get the current log types
oct	display integer in octal notation
output-file-start	send output to file
output-file-stop	stop sending output to file
output-radix	change the default output radix
pipe	run commands through a pipe
print	display integer in various bases
signed	interpret unsigned integer as signed
signed16	interpret unsigned integer as signed

<code>signed32</code>	interpret unsigned integer as signed
<code>signed64</code>	interpret unsigned integer as signed
<code>signed8</code>	interpret unsigned integer as signed
<code>undisplay</code>	remove expression installed by display

PCI

<code>create-cp3-quad100tx</code>	create a a non-instantiated cp3_quad100tx
<code>create-pci-accel-vga</code>	create a non-instantiated pci-accel-vga
<code>create-pci-am79c973</code>	create a non-instantiated pci-am79c973
<code>create-pci-bcm5703c</code>	create a non-instantiated pci-bcm5703c
<code>create-pci-bcm5704c</code>	create a non-instantiated pci-bcm5704c
<code>create-pci-dec21041</code>	create a non-instantiated pci-dec21041
<code>create-pci-dec21140a</code>	create a non-instantiated pci-dec21140a
<code>create-pci-dec21140a-dml</code>	create a non-instantiated pci-dec21140a-dml
<code>create-pci-dec21143</code>	create a non-instantiated pci-dec21143
<code>create-pci-i21152</code>	create a non-instantiated pci-i21152
<code>create-pci-i82543gc</code>	create a non-instantiated pci-i82543gc
<code>create-pci-i82546bg</code>	create a non-instantiated pci-i82546bg
<code>create-pci-i82559</code>	create a non-instantiated pci-i82559
<code>create-pci-ispl040</code>	create a non-instantiated pci-ispl040
<code>create-pci-ispl2200</code>	create a non-instantiated pci-ispl2200
<code>create-pci-pd6729</code>	create a non-instantiated pci-pd6729
<code>create-pci-sil680a</code>	create a non-instantiated pci-sil680a
<code>create-pci-sym53c810</code>	create a non-instantiated pci-sym53c810
<code>create-pci-sym53c875</code>	create a non-instantiated pci-sym53c875
<code>create-pci-sym53c876</code>	create a non-instantiated pci-sym53c876
<code>create-pci-vga</code>	create a non-instantiated pci-vga
<code>new-cp3-quad100tx</code>	create a an instantiated cp3_quad100tx

PCMCIA

<code>create-pci-pd6729</code>	create a non-instantiated pci-pd6729
<code>create-std-pcmcia-flash-disk</code>	create a non-instantiated std-pcmcia-flash-disk
<code>new-std-pcmcia-flash-disk</code>	create an instantiated std-pcmcia-flash-disk

Profiling

<code><address_profiler>.address-profile-data</code>	linear map of address profiling data
<code><address_profiler>.address-profile-info</code>	general info about an address profiler

<address_profiler>.address-profile-toplist	print toplist of address profiling data
<cycle>.print-time	print number of steps and cycles executed
<data-profiler>.clear	clear data profiler
disable-magic-breakpoint	remove breakpoint on magic break instructions
enable-magic-breakpoint	set breakpoint on magic break instructions
function-profile	list functions sorted by profile counts
instruction-fetch-mode	set or get current mode for instruction fetching
magic-breakpoint-enabled	return TRUE if the magic breakpoint is enabled
print-time	print number of steps and cycles executed
select-profiles	set profilers to display in source listing

Python

@	evaluate a Python statement
'	evaluate a Python expression
python	evaluate a Python expression
run-python-file	execute Python file

RapidIO Link

create-rapidio-link	create a a non-instantiated rapidio_link
new-rapidio-link	create a an instantiated rapidio_link

Real Network

<ethernet-link>.connect-real-network-bridge	connect to the real network
<ethernet-link>.connect-real-network-host	connect to the real network
<ethernet-link>.connect-real-network-napt	enable NAPT from simulated network
<ethernet-link>.disconnect-real-network	disconnect from the real network
<ethernet_cable>.connect-real-network-bridge	connect to the real network
<ethernet_cable>.connect-real-network-host	connect to the real network
<ethernet_cable>.connect-real-network-napt	enable NAPT from simulated network
<ethernet_cable>.disconnect-real-network	disconnect from the real network
<ethernet_hub>.connect-real-network-bridge	connect to the real network
<ethernet_hub>.connect-real-network-host	connect to the real network
<ethernet_hub>.connect-real-network-napt	enable NAPT from simulated network
<ethernet_hub>.disconnect-real-network	disconnect from the real network
<ethernet_switch>.connect-real-network-bridge	connect to the real network

3.2. By Categories

<ethernet_switch>.connect-real-network-host	connect to the real network
<ethernet_switch>.connect-real-network-napt	enable NAPT from simulated network
<ethernet_switch>.disconnect-real-network	disconnect from the real network
<ethernet_vlan_switch>.connect-real-network-napt	enable NAPT from simulated network
close-tap-interface	close an unused persistent TAP interface
connect-real-network	connect a simulated machine to the real network
connect-real-network-bridge	connect bridge between real and simulated network
connect-real-network-host	connect real host to the simulated network
connect-real-network-napt	enable NAPT from simulated network
connect-real-network-port-in	setup port forwarding to a simulated machine
connect-real-network-port-out	setup port forwarding to real machine
default-port-forward-target	set default port forwarding target
disconnect-real-network	disconnect from the real network
disconnect-real-network-port-in	remove port forwarding to a simulated machine
disconnect-real-network-port-out	remove port forwarding to real machine
list-port-forwarding-setup	view the port forwarding setup
network-helper	set/show name of host network helper

Registers

%	read register by name, module or string formatting
<int_register>.read-reg	read a register
<int_register>.register-number	<i>deprecated</i> — get the number of a processor register
<int_register>.write-reg	write to register
pregs	print cpu registers
read-reg	read a register
set-pc	set the current processor's program counter
write-reg	write to register

Reverse Execution

clear-recorder	clear recorded events
delete-bookmark	delete a time bookmark
disable-reverse-execution	disable reverse execution
enable-reverse-execution	enable reverse execution
list-bookmarks	list time bookmarks
reverse	run simulation backwards
reverse-cycles	run simulation backwards
reverse-step-instruction	reverse step one or more instruction
reverse-to	set a temporary time breakpoint and run backwards

<code>rexec-limit</code>	tune reverse execution performance parameters
<code>set-bookmark</code>	set a bookmark at the current point in time
<code>skip-to</code>	skip to the specified point in the simulation

SATA

<code>create-std-sata-cdrom</code>	create a a non-instantiated std_sata_cdrom
<code>create-std-sata-disk</code>	create a a non-instantiated std_sata_disk
<code>new-std-sata-cdrom</code>	create a an instantiated std_sata_cdrom
<code>new-std-sata-disk</code>	create a an instantiated std_sata_disk

SCSI

<code>create-pci-isp1040</code>	create a non-instantiated pci-isp1040
<code>create-pci-isp2200</code>	create a non-instantiated pci-isp2200
<code>create-pci-sym53c810</code>	create a non-instantiated pci-sym53c810
<code>create-pci-sym53c875</code>	create a non-instantiated pci-sym53c875
<code>create-pci-sym53c876</code>	create a non-instantiated pci-sym53c876
<code>create-simple-fc-disk</code>	create a non-instantiated simple-fc-disk
<code>create-std-scsi-bus</code>	create a non-instantiated std-scsi-bus
<code>create-std-scsi-cdrom</code>	create a non-instantiated std-scsi-cdrom
<code>create-std-scsi-disk</code>	create a non-instantiated std-scsi-disk
<code>new-std-scsi-bus</code>	create an instantiated std-scsi-bus

Serial Link

<code>create-ser-link</code>	create a a non-instantiated ser_link
<code>create-std-serial-link</code>	create a non-instantiated std-serial-link
<code>new-ser-link</code>	create a an instantiated ser_link
<code>new-std-serial-link</code>	create an instantiated std-serial-link

Simics Search Path

<code>add-directory</code>	add a directory to the Simics search path
<code>add-module-directory</code>	add a directory to the module search path
<code>clear-directories</code>	clear the Simics search path
<code>file-exists</code>	check if a file exists in the search path
<code>list-directories</code>	list directories in Simics search path

`resolve-file` resolve a filename

Speed

<code><cell>.cpu-switch-time</code>	get/set the time quantum for a given cell
<code><realtime>.disable</code>	disable real-time behavior
<code><realtime>.enable</code>	enable real-time behavior
<code>cpu-switch-time</code>	get/set default switch time
<code>disable-multithreading</code>	disable multithreading
<code>disable-page-sharing</code>	disable page-sharing
<code>disable-real-time-mode</code>	disable real-time behavior
<code>dsti-disable</code>	disable D-STC
<code>dsti-enable</code>	enable D-STC
<code>enable-multithreading</code>	enable multithreading
<code>enable-page-sharing</code>	enable page-sharing
<code>enable-real-time-mode</code>	enable real-time behavior
<code>iostc-disable</code>	disable IO-STC
<code>iostc-enable</code>	enable IO-STC
<code>istc-disable</code>	disable I-STC
<code>istc-enable</code>	enable I-STC
<code>new-realtime</code>	create a new realtime object
<code>set-thread-limit</code>	limit the number of simulation threads
<code>sti-status</code>	show I- and D-STC status

Status Panel

<code>create-example-status-panel</code>	create a non-instantiated example-status-panel
<code>new-example-status-panel</code>	create an instantiated example-status-panel

Symbolic Debugging

<code><context>.finish-function</code>	finish the current function
<code><context>.next-instruction</code>	run to the next instruction, skipping subroutine calls
<code><context>.next-line</code>	run to the next source line, skipping subroutine calls
<code><context>.off</code>	switch off context object
<code><context>.on</code>	switch on context object
<code><context>.reverse-next-instruction</code>	reverse to the previous instruction, skipping subroutine calls
<code><context>.reverse-next-line</code>	reverse to the previous source line, skipping subroutine calls
<code><context>.reverse-step-instruction</code>	reverse to the previous instruction

<context>.reverse-step-line	reverse to the previous source line
<context>.step-instruction	run to the next instruction
<context>.step-line	run to the next source line
<context>.syms	set the symbol table of a context
<context>.uncall-function	reverse to when the current function was called
<memory_space>.debug	get debug object
<syms>.abi	set ABI to use
<syms>.list	list source and/or disassemble
<syms>.load-symbols	load symbols from file
<syms>.plain-symbols	read raw symbols in nm format
<syms>.pos	address of line or function
<syms>.source-path	set source search path for debug info
<syms>.whereis	find symbol by address
add-pathmap-entry	add a path map entry
add-symbol-file	add symbol file to debugging contexts
clear-memorymap	clear all memory map entries
clear-pathmap	clear all path map entries
debug-context	return the current debug object
down	go down N stack frames
finish-function	finish the current function
frame	change current stack frame
list	list source and/or disassemble
list-sections	Lists the relocatable sections of a symbol file
list-segments	Lists the segments of a symbol file
new-context	create a new context
new-gdb-remote	Create a gdb session
new-remote-frontend	create a remote debugger session
new-syms	create new symbol table
new-tcf-agent	create a tcf agent
new-wdb-remote	create a wdb session
next-instruction	run to the next instruction, skipping subroutine calls
next-line	run to the next source line, skipping subroutine calls
pos	address of line or function
psym	print value of symbolic expression
reverse-next-instruction	reverse to the previous instruction, skipping subroutine calls
reverse-next-line	reverse to the previous source line, skipping subroutine calls
reverse-step-line	reverse to the previous source line
set-context	set the current context of a CPU
show-memorymap	show the current memory map
show-pathmap	show the current path map
stack-trace	display stack trace

<code>step-line</code>	run to the next source line
<code>sym-address</code>	return the address of expression or source line
<code>sym-file</code>	return source file for function or address
<code>sym-function</code>	return function at a given address
<code>sym-line</code>	return source line for function or address
<code>sym-source</code>	print source location for function or address
<code>sym-string</code>	evaluate symbolic expression
<code>sym-type</code>	return the type of a symbolic expression
<code>sym-value</code>	evaluate symbolic expression
<code>symval</code>	evaluate symbolic expression
<code>uncall-function</code>	reverse to when the current function was called
<code>up</code>	go up N stack frames
<code>whereis</code>	find symbol by address

Test

`expect` fail if not equal

Text Console

<code>create-std-host-serial-console</code>	create a non-instantiated std-host-serial-console
<code>create-std-server-console</code>	create a non-instantiated std-server-console
<code>create-std-telnet-console</code>	create a non-instantiated std-telnet-console
<code>create-std-text-console</code>	create a non-instantiated std-text-console
<code>create-std-text-graphics-console</code>	create a non-instantiated std-text-graphics-console
<code>new-std-host-serial-console</code>	create an instantiated std-host-serial-console
<code>new-std-server-console</code>	create an instantiated std-server-console
<code>new-std-telnet-console</code>	create an instantiated std-telnet-console
<code>new-std-text-console</code>	create an instantiated std-text-console
<code>new-std-text-graphics-console</code>	create an instantiated std-text-graphics-console

Tracing

<code>instruction-fetch-mode</code>	set or get current mode for instruction fetching
<code>new-tracer</code>	create a new tracer

USB

create-usb-xmas-tree	create a non-instantiated <code>usb_xmas_tree</code>
new-usb-disk-from-image	create new <code>usb</code> disk with content
new-usb-xmas-tree	create a instantiated <code>usb_xmas_tree</code>

VMP

[enable-core2-bugfix](#) enable hardware bug workaround

3.3 Global Command Descriptions

!

Synopsis

`!shell-command`

Description

Executes the rest of the command line in the system command line interpreter. Works like the `shell` command, except that the shell command does not need to be quoted.

Provided By

[Simics Core](#)

See Also

[shell](#)

!=

Synopsis

`arg1 != arg2`

Description

Returns true if `arg1` and `arg2` are not equal, and false if equal.

Provided By

[Simics Core](#)

#

Synopsis

`# comment`

Description

Single line comment.

Provided By

[Simics Core](#)

\$**Synopsis**`$var`**Description**

Gets the value of a CLI variable, like in `print $var`.

Provided By[Simics Core](#)**See Also**[defined](#), [read-variable](#)**%****Synopsis**`%reg``arg1 % arg2`**Description**

When used as a prefix on a register name, the register *reg* for the current frontend processor is accessed. This is a convenient way to use register values in expressions like **disassemble (%pc - 4*3) 10** or **%pc = 0x1000**.

The % operator is also used for arithmetic modulo of integer and floating point values.

If the argument to the left of the % operator is a string, then string formatting using Python format specifiers such as %s and %d is performed. To provide multiple values to be formatted, a CLI list should be used as right hand argument. For example:

```
simics> "about %.2f %s" % [5.12345, "hours"]
about 5.12 hours
```

Provided By[Simics Core](#)**See Also**[read-reg](#), [write-reg](#), [pregs](#), [pselect](#)**&****Synopsis**`arg1 & arg2`

Description

Bitwise AND operation.

Provided By

[Simics Core](#)

*

Synopsis

`arg1 * arg2`

Description

Arithmetic multiplication.

Provided By

[Simics Core](#)

+

Synopsis

`arg1 + arg2`

Description

Arithmetic addition, string and list concatenation.

Provided By

[Simics Core](#)

`+=`

Synopsis

`$var += value`

`%reg += value`

Description

Add a string, integer or list to a CLI variable, or an integer value to a register.

Provided By

[Simics Core](#)

-

Synopsis

`arg1 - arg2`

Description

Arithmetic subtraction.

Provided By
[Simics Core](#)

=

Synopsis

`$var -= value`
`%reg -= value`

Description

Subtract an integer from a CLI variable, or from a register.

Provided By
[Simics Core](#)

->

Synopsis

`object->attribute`
`object->attribute = expression`

Description

Access the attribute *attribute* in *object*. Only object, string, float, integer, boolean, and list attributes can be returned by the command. A nil attribute will be returned as zero.

When reading other attribute types, they will be printed but the command will not return anything.

Provided By
[Simics Core](#)

/

Synopsis

`arg1 / arg2`

Description

Arithmetic division.

Provided By
[Simics Core](#)

:

Synopsis

`component : object`

Description

Access to objects in a component should be done using the . operator instead.

Provided By

[Simics Core](#)

<

Synopsis

arg1 < *arg2*

Description

Returns true if *arg1* is less than *arg2*, and false if not.

Provided By

[Simics Core](#)

<<

Synopsis

arg1 << *arg2*

Description

Bitwise left shift.

Provided By

[Simics Core](#)

<=

Synopsis

arg1 <= *arg2*

Description

Returns true if *arg1* is less than or equal to *arg2*, and false if not.

Provided By

[Simics Core](#)

=

Synopsis

`$var = value`

`%reg = value`

Description

Sets a CLI variable to an integer, float string or list value.

Can also be used to assign an integer value to a processor register.

Provided By
[Simics Core](#)

==

Synopsis
 $\text{arg1} == \text{arg2}$

Description
Returns true if arg1 and arg2 are equal, and false if not.

Provided By
[Simics Core](#)

>

Synopsis
 $\text{arg1} > \text{arg2}$

Description
Returns true if arg1 is greater than arg2 , and false if not.

Provided By
[Simics Core](#)

>=

Synopsis
 $\text{arg1} >= \text{arg2}$

Description
Returns true if arg1 is greater than or equal to arg2 , and false if not.

Provided By
[Simics Core](#)

>>

Synopsis
 $\text{arg1} >> \text{arg2}$

Description
Bitwise right shift.

Provided By
[Simics Core](#)

@

Synopsis*@python-statement***Description**

Evaluates the rest of the command line as a Python statement and prints its result.

Provided By[Simics Core](#)**See Also**[python](#), [run-python-file](#)

[

Synopsis*\$var[idx]**\$var[idx] = value***Description**

Get or set the value for the indexed CLI variable \$var.

Provided By[Simics Core](#)

^

Synopsis*arg1 ^ arg2***Description**

Bitwise XOR operation.

Provided By[Simics Core](#)

'

Synopsis*'exp'**python "exp"***Description***exp* will be evaluated in the Python environment and the result returned to the frontend.

For example:

```
print -x 'SIM_step_count(SIM_current_processor())'
$all_objects = python "SIM_get_all_objects()"
```

Both expressions and statements can be run, but for statements the @ command can be used instead.

Provided By[Simics Core](#)**See Also**[@, run-python-file](#)**add-data-to-script-pipe****Synopsis****add-data-to-script-pipe** *pipe data***Description**

Sends data to a script branch using a pipe. The data will be queued until the receiving script branch reads it with **wait-for-script-pipe**. The data can be an integer, string, floating point value or a list.

Provided By[Simics Core](#)**See Also**[script-branch, create-script-pipe, script-pipe-has-data, wait-for-script-pipe](#)**add-directory****Synopsis****add-directory** *path [-prepend]***Description**

Adds a directory to the Simics search path. The Simics search path is a list of directories where Simics searches for additional files when loading a configuration or executing a command like **load-file**.

The value of *path* is normally appended at the end of the list. If the **-prepend** flag is given, the path will be added as first in the list.

Provided By[Simics Core](#)**See Also**[list-directories](#)**add-module-directory****Synopsis****add-module-directory** *path***Description**

Adds a directory to the Simics module search path. This path is used to look for additional modules, that can be used to extend the functionality of Simics.

Provided By
[Simics Core](#)

add-pathmap-entry

Synopsis

add-pathmap-entry “*source*” *destination* [“*context-query*”]

Description

Add an entry mapping from paths in binaries known by the debugger to paths on the Simics host. The mapping can be limited to debug contexts matching a particular context query with *context-query*. It defaults to *, which matches all debug contexts.

The pathmap is the name for the translations from paths in target binaries to paths on the Simics host. This pathmap is specific to the command line interface to the debugger, it is not shared with the Eclipse frontend to the debugger.

Provided By
[tcf-agent](#)

See Also

[show-pathmap](#)

add-symbol-file

Synopsis

add-symbol-file *symbolfile* [“*context-query*”] [*relocation-address*] [“*section*”] [*segment*]

Description

Add the symbol file *symbolfile* to debugging contexts matching the context query *context-query*.

symbolfile uses Simics’s Search Path and path markers (%simics%, %script%) to find the symbol file. Refer to *The Command Line Interface* chapter of the *Hindsight User’s Guide* manual for more information on how Simics’s Search Path is used to locate files.

context-query defaults to * which matches all contexts.

relocation-address is the address, the file/segment/section should be mapped to.

section is the name of the section to relocate. See the **list-sections** command.

segment is the number of the segment to relocate. See the **list-segments** command.

Provided By
[tcf-agent](#)

See Also

[list-sections](#), [list-segments](#), [show-memorymap](#)

alias

Synopsis

```
alias ["alias"] ["as"] [-r]
```

Description

Make an alias for a command or an object reference. The *alias* argument is the name of the new alias and the *as* argument the command or reference to make an alias for. For example:

```
alias ds disassemble-settings
```

will create an alias 'ds' for the command 'disassemble-settings'. So instead of writing disassemble-setting you can write 'ds' at the command line or in a Simics script.

You can also define an alias for an object reference like:

```
alias cpu0 system_cmp0.board1.cpu[0]
```

which allows you to write, for example, 'cpu0.pregs' instead of 'system_cmp0.board1.cpu[0].pregs' to print the registers for the cpu[0] processor in the component board1 in the component system_cmp0.

Use **alias** *alias as -r* if you want to replace an existing alias with a new one, or **alias** *alias -r* if you want to remove the alias.

Aliases defined by this command have higher priority than build-in aliases. This allows you to redefine the built-ins.

If no arguments are given a list of all aliases, including the built-ins, will be printed.

Aliases must only contain letters, digits and underscores. The first character is not allowed to be a digit.

Provided By

[Simics Core](#)

and

Synopsis

```
arg1 and arg2
```

Description

Evaluates *arg1* and returns its value if it is false. Otherwise *arg2* is evaluated and its value is returned.

Provided By

[Simics Core](#)

api-help

Synopsis

```
api-help "topic"
```

Description

Shows API help on topic *topic*.

For C API functions or data types, shows only the declaration. See the *Simics Reference Manual* for further detail.

For Python API functions or classes, also shows the documentation.

This command does the same thing as **help api:*topic***.

Provided By

[Simics Core](#)

See Also

[help](#), [api-search](#), [help-search](#), [win-help](#)

api-search**Alias**

[api-apropos](#)

Synopsis

api-search “*search-string*”

Description

Search the API documentation for the string *search-string*.

Provided By

[Simics Core](#)

See Also

[api-help](#), [help](#), [help-search](#)

auto-partition-configuration**Synopsis**

auto-partition-configuration [*checkpoint*]

Description

Print a suggested partitioning of the configuration objects in the given checkpoint into separate simulation cells. This suggestion is based on the actual connections between the objects.

If no checkpoint is given, check the currently loaded configuration.

Provided By

[Simics Core](#)

bin

Synopsis

```
bin value [-u] [-p]
```

Description

Returns the parameter as a string in binary notation. This is similar to **print -b** *value*. To ignore any default digit grouping, the **-u** (unformatted) flag is used, while **-p** removes the radix prefix 0b.

Provided By

[Simics Core](#)

See Also

[print](#), [hex](#), [oct](#), [dec](#), [digit-grouping](#)

break**Alias**

b

Synopsis

```
break address [length] [-r] [-w] [-x]
<breakpoint>.break address [length] [-r] [-w] [-x]
<breakpoint>.tbreak address [length] [-r] [-w] [-x]
```

Description

Add breakpoint (read, write, or execute) on an object implementing the breakpoint interface. This is typically a memory space object such as physical memory; e.g., **phys_mem0.break 0xff3800**. Accesses intersecting the given range will trigger the breakpoint. By default the breakpoint will only trigger for instruction execution, but any subset of read, write, and execute accesses can be set to trigger using combinations of **-r**, **-w**, and **-x**.

length is the interval length in bytes (default is 1).

Breakpoints inserted with the **tbreak** command are automatically disabled when they have triggered.

The default action at a triggered breakpoint is to return to the frontend. This can be changed by using haps. When an execution breakpoint is triggered, Simics will return to the command prompt before the instructions is executed, while instructions triggering read or write breakpoints will complete before control is returned to the command prompt.

To break on a virtual address, use a context object:

```
cpu0_context.break 0x1ff00
```

Several breakpoints can be set on the same address and Simics will break on them in turn. If hap handlers (callback functions) are connected to the breakpoints they will also be executed in turn. Hap handlers are called before the access is performed, allowing

the user to read a memory value that may be overwritten by the access. See the Simics Reference Manual for a description of hap handlers.

Each breakpoint is associated with an id (printed when the breakpoint is set or by the **list-breakpoints** command) which is used for further references to the breakpoint.

For convenience there are also a **break** command which sets a breakpoint on memory connected to the current frontend processor (see **pselect**). Default is to break on virtual address accesses (in the current context). By prefixing the address with *p*: it is possible to break on physical accesses as well (cf. **phys_mem0.break**); e.g., **break p:0xffc0**.

Several attributes can be set for a breakpoint for breaking only when some conditions are true. See the **disable**, **enable**, **ignore**, **set-prefix**, **set-substr** and **set-pattern** commands for more details.

Breakpoints can be removed using **delete**.

Provided By

[Simics Core](#)

See Also

[unbreak](#), [delete](#), [enable](#), [ignore](#), [set-prefix](#), [set-substr](#), [set-pattern](#), [list-breakpoints](#), [wait-for-breakpoint](#)

break-cr

Synopsis

```
break-cr ("register"|-all|-list)
<int_register>.break-cr ("register"|-all|-list)
<int_register>.unbreak-cr ("register"|-all|-list)
unbreak-cr ("register"|-all|-list)
```

Description

Enables and disables breaking simulation on control register updates. When this is enabled, every time the specified control register is updated during simulation a message is printed. The message will name the register being updated, and the new value. The new value will be printed even if it is identical to the previous value.

The *reg-name* parameter specifies which control register should be traced. The available control registers depends on the simulated target.

Instead of a register name, the *-all* flag may be given. This will enable or disable tracing of all control register.

Using the *-list* argument will print out the registers accesses on which a breakpoint will trigger.

The non-namespace variant of this command breaks the simulation on control register updates on all processors of the same class in the same group as the currently injected processor. The namespace variant affects only the selected processor.

Provided By

[Simics Core](#)

See Also

[trace-cr](#), [`<breakpoint>.break`](#), [log-setup](#)

break-exception**Synopsis**

break-exception (*number*|“*name*”|-all|-list)

Description

Enables and disables breaking simulation on exceptions. When this is enabled, every time the specified exception occurs during simulation a message is printed.

The *exception* parameter specifies which exception should be traced. The available exceptions depends on the simulated target.

Instead of an exception, the `-all` flag may be given. This will enable or disable tracing of all exceptions.

Using the `-list` argument will print out the exceptions on which a breakpoint will trigger.

The non-namespace variant of this command installs breakpoints on all processors of the same class in the same group as the currently inspected processor. The namespace variant affects only the selected processor.

Provided By

[Simics Core](#)

See Also

[trace-exception](#), [`<breakpoint>.break`](#)

break-hap**Synopsis**

break-hap (“*hap*”|-all|-list)
unbreak-hap (“*hap*”|-all|-list)

Description

Enables and disables breaking simulation on haps. When this is enabled, every time the specified hap is triggered a message is printed and simulation is stopped.

The *hap* parameter specifies the hap.

Instead of a hap, the `-all` flag may be given. This will enable or disable breaking on all haps.

Using the `-list` argument will print out the haps on which a breakpoint will trigger.

Provided By

[Simics Core](#)

See Also[trace-hap](#), [list-haps](#)**break-io****Synopsis**

break-io (“*device*”|-all|-list)
unbreak-io (“*device*”|-all|-list)

Description

Enables and disables breaking simulation on device accesses. When this is enabled, every time the specified device is accessed during simulation a message is printed and the simulation stops.

The *device* parameter specifies which device object should be traced.

Instead of an object name, the `-all` flag may be given. This will enable or disable breaking on accesses to all device.

Using the `-list` argument will print out the objects for which I/O breakpoints will trigger.

Provided By[Simics Core](#)**See Also**[trace-io](#), <breakpoint>.break**break-line****Synopsis**

break-line “*line*” [“*context-query*”]

Description

Add a breakpoint at a given source code line. The *line* should be given as file-name:linenumber[:columnnumber].

With the `-x`, `-r`, `-w` flags you can specify if the breakpoint should be an execution breakpoint, a read breakpoint, a write breakpoint, or any combination. The default is an execution breakpoint.

You can limit the debug contexts the breakpoint applies to by providing a context query with *context-query*. It defaults to `*`, which matches all debug context.

The command returns the *id* of the new breakpoint. This can be used to manage and delete the breakpoint using the **breakpoints** object.

Provided By[tcf-agent](#)**See Also**[break-location](#)

break-location**Synopsis**

```
break-location ("location" | address) [length] [-r] [-w] [-x] ["context-query"]
```

Description

Add a breakpoint on a particular location. The location can be specified either as a memory address or a C expression. In addition you can specify the size of the breakpoint, i.e. how many consecutive memory addresses it should match with *size*.

With the *-x*, *-r*, *-w* flags you can specify if the breakpoint should be an execution breakpoint, a read breakpoint, a write breakpoint, or any combination. The default is an execution breakpoint.

You can limit the debug contexts the breakpoint applies to by providing a context query with *context-query*. It defaults to ***, which matches all debug context.

The command returns the *id* of the new breakpoint. This can be used to manage and delete the breakpoint using the **breakpoints** object.

Provided By

[tcf-agent](#)

See Also

[break-line](#)

break-log**Synopsis**

```
break-log ["substring"] [object] [type]
```

Description

Break on log message. With no arguments the simulation will stop when the next log message is printed. By specifying *object*, *type*, and / or *substring* it will stop on the next log message matching these conditions. The break is triggered once only, to break again you need to run the command again.

Provided By

[Simics Core](#)

cd**Synopsis**

```
cd path
```

Description

Change working directory of Simics. Works as if you had done 'cd' at the shell. Converts *param* to host native form first (see **native-path**).

Provided By
[Simics Core](#)

See Also
[ls](#), [pwd](#)

change-namespace

Alias
 cn

Synopsis
change-namespace [“*component*”]

Description

Change current namespace to the *component*. This means that you do not need to provide the full path when you refer to objects within the component. You can still refer to top level components.

For example if you have a component called *system_cmp0* with a slot *cpu* containing a processor, you can do:

```
cn system-cmp0
cpu.log-level 2
```

If there are nested components you can also change the current namespace by providing the slot name of component to the change-namespace command.

Use .. to go “up” one level. To go to a namespace relative to the top level, prefix it with a dot:

```
cn .system-cmp0
```

If no component is given, no current namespace will be set.

If a current namespace is set the **list-object** command will only list the objects in that namespace.

Provided By
[Simics Core](#)

See Also
[current-namespace](#), [list-objects](#)

check-cell-partitioning**Synopsis**

check-cell-partitioning [*checkpoint*]

Description

Verify that the cell partitioning of the given checkpoint into cells is correct; that is, that no objects in one cell have references to objects in another cell.

If no checkpoint is given, check the currently loaded configuration.

Provided By

[Simics Core](#)

clear-directories**Synopsis**

clear-directories

Description

Empty the Simics search path.

Provided By

[Simics Core](#)

clear-memorymap**Synopsis**

clear-memorymap

Description**Provided By**

[tcf-agent](#)

clear-pathmap**Synopsis**

clear-pathmap

Description

Remove all entries used to translate from paths in binaries to paths on the Simics host.

The pathmap is the name for the translations from paths in target binaries to paths on the Simics host. This pathmap is specific to the command line interface to the debugger, it is not shared with the Eclipse frontend to the debugger.

Provided By

[tcf-agent](#)

clear-recorder**Synopsis****clear-recorder****Description**

Discard recorded input events (e.g. human console input). This command allows an alternate future to take place.

Provided By[Simics Core](#)**See Also**[simulation-reversing](#)**close-tap-interface****Synopsis****close-tap-interface** [*"interface"*]**Description**

Closes a TAP interface that has been set persistent. If the interface is currently used, it will not be closed at once, only the persistent state will change. Not applicable for for TAP-Win32 interfaces.

Provided By[real-network](#)**command-history****Synopsis****command-history** [*max-lines*]**Description**

Prints the recent CLI command-line history. Up to *max-lines* lines are displayed, default is 100.

Provided By[Simics Core](#)**command-list****Synopsis****command-list** *file***Description**

Produces a quick reference list of Simics commands in HTML. Note that Simics commands can be added at runtime, and any particular dynamic usage will be different from printed manuals. *file* is the file to write to.

Provided By
[Simics Core](#)

configuration-shortest-paths

Synopsis

configuration-shortest-paths “*object_A*” “*object_B*” [*checkpoint*]

Description

Print all the shortest paths between two objects in the given checkpoint, or in the currently loaded configuration if no checkpoint is given.

Provided By
[Simics Core](#)

connect

Synopsis

connect [*cnt0*] [*cnt1*]

Description

Connect connector *cnt0* to *cnt1*.

Provided By
[Simics Core](#)

connect-central

Synopsis

connect-central [“*server*”|*obj*] [*reconnect*]

Description

Connect Simics to a Simics Central server.

The *server* argument specifies the server to connect to. It is either of the form <addr>[:<port>] if a TCP connection should be used, or a file name if a file socket should be used. This is the same as the command line argument –central, but can be executed at any time.

To connect to a server object in the same Simics, use the *obj* argument instead.

If neither *server* nor *obj* is specified, it tries to find a central server in the same process or on the same host.

The *reconnect* argument gives the time between tries to connect to the server if a connection can't be established, or if the connection is lost. A value of zero disables reconnection. The default is to not try to reconnect.

Provided By
[central](#)

See Also

[new-central-server](#), [<central-client>.disconnect](#), [<central-client>.connect](#)

connect-components — deprecated**Synopsis**

connect-components *src-component* [“*src-connector*”] [*component*] [“*dst-connector*”] [-f]

Description

This command is deprecated; use ‘[connect <cnt0> <cnt1>](#)’ instead.

Creates a connection between the connectors *src-connector* and *dst-connector* of the *src-component* and *dst-component* components. Indexed connectors must be quoted. The *-f* flag selects the first unused connector (in alphabetic order).

Provided By

[Simics Core](#)

connect-panel-to-frontend**Synopsis**

connect-panel-to-frontend *panel* *frontend*

Description**Provided By**

[system-panel](#)

connect-real-network**Synopsis**

connect-real-network [“*target-ip*”] [*ethernet-link*] [“*service-node-ip*”]

Description

Enables port forwarding from the host that Simics is running on, to a simulated machine specified by *target-ip*, allowing access from real hosts to the simulated one.

Ports are opened on the host for a number of commonly used protocol (such as FTP and telnet). Additional ports can be configured using the **connect-real-network-port-in** command.

Port forwarding can be enabled for several simulated machines at the same time. This command also enables NAPT for accesses that are initiated from the simulated machine.

If several Ethernet links exists, the one that the simulated machine is connected to must be specified. If no Ethernet link exists, one will be created and all Ethernet devices are connected to it.

A **service-node** will also be added to the link if there isn't one connected already. If a service-node is added it will either get the IP address *service-node-ip*, if it was specified, or the IP of the target with the lowest byte set to 1.

If a default port-forwarding target has been set using the **default-port-forward-target** command, then the *target-ip* argument can be left out.

Provided By
[service-node](#)

See Also

[default-port-forward-target](#), [connect-real-network-napt](#), [connect-real-network-port-in](#), [connect-real-network-port-out](#), [connect-real-network-host](#), [connect-real-network-bridge](#)

connect-real-network-bridge

Synopsis

```
connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent]
[-propagate-link-status]
<ethernet-link>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-
xlate] [-persistent] [-propagate-link-status]
<ethernet_cable>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-
xlate] [-persistent] [-propagate-link-status]
<ethernet_hub>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-
xlate] [-persistent] [-propagate-link-status]
<ethernet_switch>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-
xlate] [-persistent] [-propagate-link-status]
```

Description

Creates an Ethernet bridge between a simulated Ethernet link and a real network through an Ethernet interface of the simulation host.

The optional *interface* argument specifies the Ethernet or TAP interface of the host to use.

By default a TAP interface is used, but if the *host-access* argument is `raw`, raw access to an Ethernet interface is used.

MAC address translation can be disabled with the `-no-mac-xlate` flag.

The `-persistent` is for backward compatibility and should not be used.

If `-propagate-link-status` is specified, link status changes on the host interface will be propagated to all devices on the link that implements the link-status interface. For TAP, only 'up' and 'down' status will be propagated (and not 'unconnected'). Link status propagation is only supported on Linux.

This command returns the new real-network component object.

See the *Connecting to a Real Network* chapter of the *Ethernet Networks in Simics* manual for more information about how to connect to a real network.

Provided By
[real-network](#)

See Also
[connect-real-network](#), [connect-real-network-host](#), [disconnect-real-network](#)

connect-real-network-host

Synopsis

```
connect-real-network-host ["interface"] [-persistent]
<ethernet-link>.connect-real-network-host ["interface"] [-persistent]
<ethernet_cable>.connect-real-network-host ["interface"] [-persistent]
<ethernet_hub>.connect-real-network-host ["interface"] [-persistent]
<ethernet_switch>.connect-real-network-host ["interface"] [-persistent]
```

Description

Connects a TAP interface of the simulation host to a simulated Ethernet link.

The optional *interface* argument specifies the TAP interface of the host to use.

The *-persistent* is for backward compatibility and should not be used.

This command returns the new real-network component object.

See the *Connecting to a Real Network* chapter of the *Simics User Guide* for more information about how to connect to a real network.

Provided By
[real-network](#)

See Also
[connect-real-network](#), [connect-real-network-bridge](#), [disconnect-real-network](#)

connect-real-network-napt

Synopsis

```
connect-real-network-napt ethernet-link [service-node]
<ethernet-link>.connect-real-network-napt [service-node]
<ethernet_cable>.connect-real-network-napt [service-node]
<ethernet_hub>.connect-real-network-napt [service-node]
<ethernet_switch>.connect-real-network-napt [service-node]
<ethernet_vlan_switch>.connect-real-network-napt [service-node]
```

Description

Enables machines on the simulated network to initiate accesses to real hosts without the need to configure the simulated machine with a real IP address. NAPT (Network Address Port Translation) uses the IP address and a port number of the host that Simics is running on to perform the access. Replies are then translated back to match the request from the simulated machine. This command also enables NAPT for accesses that are initiated from the simulated machine.

Provided By
[service-node](#)

See Also

[connect-real-network](#), [connect-real-network-port-in](#), [connect-real-network-port-out](#),
[connect-real-network-host](#), [connect-real-network-bridge](#)

connect-real-network-port-in**Synopsis**

connect-real-network-port-in (*target-port*|“*service*”) *ethernet-link* [*service-node*] [“*host-ip*”] [*host-port*] [“*target-ip*”] [-tcp] [-udp] [-f]
disconnect-real-network-port-in (*target-port*|“*service*”) *ethernet-link* [*service-node*] [“*host-ip*”] [*host-port*] [“*target-ip*”] [-tcp] [-udp]

Description

Enables or disables port forwarding from the host that Simics is running on, to a simulated machine, specified by *target-ip*. The externally visible port *host-port* on the host is mapped to the port *target-port* on the simulated machine. For commonly used services the string argument *service* can be used instead of a port number. If several Ethernet links exists, the one that the simulated machine is connected to must be specified.

The flags *-tcp* and *-udp* can be used to specify the protocol to forward. The default is to forward only the usual protocol for named services and both *tcp* and *udp* for numerically specified ports.

The flag *-f* can be used to cause the command to fail if the suggested host port could not be allocated, without the flag the command will assign the first available port starting from the specified host port and upwards.

The *host-port* given is only a hint, and the actual port used may be a different one. The command output shows the actual port used, and it can also be determined by inspecting the connections attribute in the appropriate port forwarding object.

Provided By
[service-node](#)

See Also

[connect-real-network](#), [connect-real-network-port-out](#), [connect-real-network-napt](#), [connect-real-network-host](#), [connect-real-network-bridge](#)

connect-real-network-port-out**Synopsis**

connect-real-network-port-out *service-node-port* *ethernet-link* [*service-node*] “*target-ip*”
target-port [-tcp] [-udp]
disconnect-real-network-port-out *service-node-port* *ethernet-link* [*service-node*] “*target-ip*”
target-port [-tcp] [-udp]

Description

Enables port forwarding to a machine on the real network. Traffic targeting port *service-node-port* on the service node connected to *ethernet-link* will be forwarded to port *target-port* on *target-ip*.

Both *tcp* and *udp* will be forwarded unless one of the *-tcp* or *-udp* flags are given in which case only that protocol will be forwarded.

Provided By

[service-node](#)

See Also

[connect-real-network](#), [connect-real-network-port-in](#), [connect-real-network-napt](#), [connect-real-network-host](#), [connect-real-network-bridge](#)

connect-real-network-router — deprecated**Synopsis**

```
connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]
<ethernet-link>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]

<ethernet_cable>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]

<ethernet_hub>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]

<ethernet_switch>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]
```

Description

This command is deprecated; use [connect-real-network-bridge](#) or [connect-real-network-host](#) instead.

Deprecated command

Provided By

[real-network](#)

See Also

[connect-real-network](#), [connect-real-network-host](#), [connect-real-network-bridge](#), [disconnect-real-network](#)

copy-connector**Synopsis**

```
copy-connector src "dst"
```

Description

Copy connector from *src* to *dst*.

Provided By
[Simics Core](#)

copyright

Synopsis
copyright

Description

Prints the copyright information that applies to this copy of Simics.

Provided By
[Simics Core](#)

See Also

[license](#)

cpu-switch-time

Synopsis
cpu-switch-time [*cycles|seconds*]

Description

Change the time, in cycles or seconds, between CPU switches. This command sets the CPU switch time globally. All existing cells will have their time quantum reset to the new value, and new cells will use the new value when created. Without argument, this command prints the time quantum of all cells in the system, as well as the default time quantum (if set). If the argument is given in cycles the *time_quantum* attribute is unset, and if the argument is given in seconds the *cpu_switch_time* attribute is unset, both in the **sim** class.

Provided By
[Simics Core](#)

See Also

[`<cell>.cpu-switch-time`](#), [`<sim>.time_quantum`](#), [`<sim>.cpu_switch_time`](#)

create-and-connect-ddr-memory

Synopsis
create-and-connect-ddr-memory “*system*” *memory_megs* “*organization*” [“*slot_name*”]
[*ranks_per_module*] [*min_module_size*] [*max_module_size*] [*ecc*] [“*ddr_type*”] [“*module_type*”] [-h] [*columns*] [*rows*]

Description

Create and connect DDR memory modules to the *system*. The *memory_megs* attribute defines the total module memory size in MB. It is possible to create different kind of module combinations with the *organization* parameter. The *organization* is a string. Each character in the string represents a module. The first character is module 0, second character is module 1, etc. Supported characters are a-d and A-D. Two modules can have the same character. An equal upper case character means that the modules must be of identical size. An equal lower case character means that the modules must be identical or unpopulated. The character '-' indicates that the slot should not contain any modules. A maximum of 8 characters are supported, i.e. maximum number of DIMMs is 8.

Example 1: AB Create one or two modules with any size (all slots need not be populated).

Example 2: AA Create two modules with identical size.

Example 3: aa Create one module or two modules with identical size.

Optional arguments are *ranks_per_module*, *min_module_size*, *max_module_size*, and *ecc*.

The *-h* flag will put the created memory component in the slot named *memoryX* in *system*, where X is the DIMM number.

Provided By

[memory-components](#)

create-cell-and-clocks**Synopsis**

`create-cell-and-clocks ["name"] [clock_number] [freq_mhz] [domain]`

Description

Creates a non-instantiated component of the class "cell-and-clocks". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[clock](#)

create-cp3-quad100tx**Synopsis**

`create-cp3-quad100tx ["name"] "mac_addr1" "mac_addr2" "mac_addr3" "mac_addr4"`

Description

This command creates a non-instantiated component of the class **cp3_quad100tx**.

The class description for the **cp3_quad100tx** class: The cp3-quad100tx component class.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***mac_addr1* is Required**

The MAC address of eth1

***mac_addr2* is Required**

The MAC address of eth2

***mac_addr3* is Required**

The MAC address of eth3

***mac_addr4* is Required**

The MAC address of eth4

Provided By

[cp3_quad100tx](#)

create-datatype-link**Synopsis**

create-datatype-link [“*name*”] [“*global_id*”] [*goal_latency*]

Description

This command creates a non-instantiated component of the class **datatype_link**.

The class description for the **datatype_link** class: The datatype link component creates a datatype-link, which is a simple broadcast bus forwarding messages (as sequences of bytes) from a sender device to all other devices present of the link. The datatype-link is both an example of how to build a link with the Simics Link Library, and a simple broadcast link that can be reused when multi-cell communication between devices is necessary. Refer to the *Link Library Programming Guide* for more information.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***global_id* is Optional**

Global identifier for use in distributed simulation or NIL if the link is not distributed.

***goal_latency* is Optional**

Goal latency in seconds for this link. See also the **set-min-latency** command.

Provided By

[datatype-link](#)

create-ddr-memory-module**Synopsis**

create-ddr-memory-module [“*name*”] [*rows*] [*columns*] [*ranks*] [*module_data_width*] [*primary_width*] [*ecc_width*] [*banks*] [*rank_density*] [“*module_type*”] [*cas_latency*] [“*speed*”]

Description

Creates a non-instantiated component of the class “ddr-memory-module”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[memory-components](#)

create-ddr2-memory-module**Synopsis**

```
create-ddr2-memory-module ["name"] [rows] [columns] [ranks] [module_data_width]
[primary_width] [ecc_width] [banks] [rank_density] ["module_type"] [cas_latency]
```

Description

Creates a non-instantiated component of the class “ddr2-memory-module”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[memory-components](#)

create-ddr3-memory-module**Synopsis**

```
create-ddr3-memory-module ["name"] [rows] [columns] [ranks] [module_data_width]
[primary_width] [ecc_width] [banks] [rank_density] ["module_type"] [cas_latency]
```

Description

Creates a non-instantiated component of the class “ddr3-memory-module”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[memory-components](#)

create-dummy-component**Synopsis**

```
create-dummy-component ["name"] [domain]
```

Description

Creates a non-instantiated component of the class “dummy-component”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

create-etg-comp**Synopsis**

```
create-etg-comp ["name"] "dst_ip" ["gateway_ip"] "ip" ["mac_address"] "netmask" [packet_size] [port] [pps]
```

Description

This command creates a non-instantiated component of the class **etg_comp**.

The class description for the **etg_comp** class: The etg component represents an Ethernet traffic generator.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***dst_ip* is Required**

Destination IP address for generated traffic.

***gateway_ip* is Optional**

'Gateway for non-local traffic.

***ip* is Required**

IP address of the traffic generator.

***mac_address* is Optional**

The MAC address of the traffic generator.

***netmask* is Required**

IP netmask of the traffic generator.

***packet_size* is Optional**

Packet size.

***port* is Optional**

Port.

***pps* is Optional**

Traffic rate in packets per second.

Provided By

[std-components](#)

create-etg-panel**Synopsis**

```
create-etg-panel ["name"]
```

Description

This command creates a non-instantiated component of the class **etg_panel**.

The class description for the **etg_panel** class: The Ethernet Generator System Panel.

***name* is Optional**

If not specified, the component will get a class-specific default name.

Provided By
[std-components](#)

create-eth-injector-comp

Synopsis
create-eth-injector-comp [“*name*”]

Description

This command creates a non-instantiated component of the class **eth_injector_comp**.

The class description for the **eth_injector_comp** class: The Ethernet frame injector is a pseudo-device that reads a pcap formatted file and inject the packets it found into another device, or an Ethernet link.

***name* is Optional**

If not specified, the component will get a class-specific default name.

Provided By
[eth_injector_comp](#)

create-ethernet-cable

Synopsis
create-ethernet-cable [“*name*”] [“*global_id*”] [*goal_latency*]

Description

This command creates a non-instantiated component of the class **ethernet_cable**.

The class description for the **ethernet_cable** class: Ethernet cable: this component represents a two-points ethernet cable, allowing two devices to connect to each other.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***global_id* is Optional**

Global identifier for use in distributed simulation or NIL if the link is not distributed.

***goal_latency* is Optional**

Goal latency in seconds for this link. See also the **set-min-latency** command.

Provided By
[eth-links](#)

create-ethernet-hub

Synopsis
create-ethernet-hub [“*name*”] [“*global_id*”] [*goal_latency*]

Description

This command creates a non-instantiated component of the class **ethernet_hub**.

The class description for the **ethernet_hub** class: Ethernet hub: this component represents a simple broadcasting ethernet link allowing any number of devices to connect.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***global_id* is Optional**

Global identifier for use in distributed simulation or NIL if the link is not distributed.

***goal_latency* is Optional**

Goal latency in seconds for this link. See also the **set-min-latency** command.

Provided By

[eth-links](#)

create-ethernet-switch**Synopsis**

create-ethernet-switch [*"name"*] [*"global_id"*] [*goal_latency*]

Description

This command creates a non-instantiated component of the class **ethernet_switch**.

The class description for the **ethernet_switch** class: Ethernet switch: this component represents a switched ethernet network, allowing any number of devices to connect and optimizing the packet routing according to what is learned about the MAC addresses talking on the link.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***global_id* is Optional**

Global identifier for use in distributed simulation or NIL if the link is not distributed.

***goal_latency* is Optional**

Goal latency in seconds for this link. See also the **set-min-latency** command.

Provided By

[eth-links](#)

create-ethernet-vlan-switch**Synopsis**

create-ethernet-vlan-switch [*"name"*] [*"global_id"*] [*goal_latency*]

Description

This command creates a non-instantiated component of the class **ethernet_vlan_switch**.

The class description for the **ethernet_vlan_switch** class: Ethernet VLAN switch: this component represents a switched ethernet network with VLAN support. Any number of devices is allowed to connect to various ports of the switch. Each port can be configured with its own VLAN information, in order to create sub-networks in the switch.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***global_id* is Optional**

Global identifier for use in distributed simulation or NIL if the link is not distributed.

***goal_latency* is Optional**

Goal latency in seconds for this link. See also the **set-min-latency** command.

Provided By

[eth-links](#)

create-example-keypad**Synopsis**

create-example-keypad [*"name"*]

Description

Creates a non-instantiated component of the class “example-keypad”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

create-example-status-panel**Synopsis**

create-example-status-panel [*"name"*]

Description

Creates a non-instantiated component of the class “example-status-panel”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

create-generic-pcie-switch**Synopsis**

```
create-generic-pcie-switch [“name”] [device_id] [port_count] [vendor_id]
```

Description

This command creates a non-instantiated component of the class **generic_pcie_switch**.

The class description for the **generic_pcie_switch** class: Generic PCIe switch with a configurable number of ports and configurable vendor and device IDs

***name* is Optional**

If not specified, the component will get a class-specific default name.

***device_id* is Optional**

Device ID

***port_count* is Optional**

Number of down ports

***vendor_id* is Optional**

Vendor ID

Provided By

[generic-pcie-switch-comp](#)

create-i2c-link-v2**Synopsis**

```
create-i2c-link-v2 [“name”] [“global_id”] [goal_latency]
```

Description

This command creates a non-instantiated component of the class **i2c_link_v2**.

The class description for the **i2c_link_v2** class: This component represents a simple i2c link allowing any number of devices to connect.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***global_id* is Optional**

Global identifier for use in distributed simulation or NIL if the link is not distributed.

***goal_latency* is Optional**

Goal latency in seconds for this link. See also the **set-min-latency** command.

Provided By

[i2c-link-v2](#)

create-instruction-data-splitter

Synopsis

```
create-instruction-data-splitter ["name"]
```

Description

Creates a non-instantiated component of the class “instruction-data-splitter”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[timing-components](#)

create-isa-fourport**Synopsis**

```
create-isa-fourport ["name"] [irq_level] [base_port]
```

Description

Creates a non-instantiated component of the class “isa-fourport”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[isa-components](#)

create-isa-lance**Synopsis**

```
create-isa-lance ["name"] "mac_address" [irq_level] [base_port]
```

Description

Creates a non-instantiated component of the class “isa-lance”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[isa-components](#)

create-isa-vga**Synopsis**

```
create-isa-vga ["name"] ["bios"]
```

Description

Creates a non-instantiated component of the class “isa-vga”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[isa-components](#)

create-memory-timer

Synopsis
create-memory-timer [“*name*”] [*stall_time*]

Description

Creates a non-instantiated component of the class “memory-timer”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[timing-components](#)

create-micron-mtfc2ggqdi-emmc-card

Synopsis
create-micron-mtfc2ggqdi-emmc-card [“*name*”] [“*file*”]

Description

This command creates a non-instantiated component of the class **micron_mtfc2ggqdi_emmc_card**.

The class description for the **micron_mtfc2ggqdi_emmc_card** class: A 2GB Micron eMMC card.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***file* is Optional**

File with disk contents for the full disk. Either a raw file or a CRAFF file

Provided By
[mmc-card-components](#)

create-micron-mtfc4ggqdi-emmc-card

Synopsis
create-micron-mtfc4ggqdi-emmc-card [“*name*”] [“*file*”]

Description

This command creates a non-instantiated component of the class **micron_mtfc4ggqdi_emmc_card**.

The class description for the **micron_mtfc4ggqdi_emmc_card** class: A Micron MTFC4GGQDI-IT eMMC Card.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***file* is Optional**

File with disk contents for the full disk. Either a raw file or a CRAFF file

Provided By

[mmc-card-components](#)

create-micron-mtfc4ggqdi-sdhc-card**Synopsis**

create-micron-mtfc4ggqdi-sdhc-card [“*name*”] [“*file*”]

Description

This command creates a non-instantiated component of the class **micron_mtfc4ggqdi_sdhc_card**.

The class description for the **micron_mtfc4ggqdi_sdhc_card** class: A SDHC card component similar to Micron MTFC4GGQDI-IT eMMC.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***file* is Optional**

File with disk contents for the full disk. Either a raw file or a CRAFF file

Provided By

[mmc-card-components](#)

create-os-awareness**Synopsis**

create-os-awareness [“*name*”]

Description

This command creates a non-instantiated component of the class **os_awareness**.

The class description for the **os_awareness** class: The “os-awareness” component tracks software running on the system.

***name* is Optional**

If not specified, the component will get a class-specific default name.

Provided By

[os-awareness](#)

create-pc-dual-serial-ports

Synopsis

```
create-pc-dual-serial-ports [“name”]
```

Description

Creates a non-instantiated component of the class “pc-dual-serial-ports”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[isa-components](#)

create-pc-floppy-controller**Synopsis**

```
create-pc-floppy-controller [“name”]
```

Description

Creates a non-instantiated component of the class “pc-floppy-controller”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[isa-components](#)

create-pc-quad-serial-ports**Synopsis**

```
create-pc-quad-serial-ports [“name”]
```

Description

Creates a non-instantiated component of the class “pc-quad-serial-ports”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[isa-components](#)

create-pc-single-parallel-port**Synopsis**

```
create-pc-single-parallel-port [“name”]
```

Description

Creates a non-instantiated component of the class “pc-single-parallel-port”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[isa-components](#)

create-pci-accel-vga

Synopsis
create-pci-accel-vga [“*name*”]

Description

Creates a non-instantiated component of the class “pci-accel-vga”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[pci-components](#)

create-pci-am79c973

Synopsis
create-pci-am79c973 [“*name*”] “*mac_address*” [“*bios*”]

Description

Creates a non-instantiated component of the class “pci-am79c973”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[pci-components](#)

create-pci-bcm5703c

Synopsis
create-pci-bcm5703c [“*name*”] “*mac_address*” [“*bios*”]

Description

Creates a non-instantiated component of the class “pci-bcm5703c”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[pci-components](#)

create-pci-bcm5704c

Synopsis
create-pci-bcm5704c [“*name*”] “*mac_address0*” “*mac_address1*” [“*bios*”]

Description

Creates a non-instantiated component of the class “pci-bcm5704c”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[pci-components](#)

create-pci-dec21041**Synopsis**

`create-pci-dec21041 [“name”] “mac_address” [“bios”]`

Description

Creates a non-instantiated component of the class “pci-dec21041”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[pci-components](#)

create-pci-dec21140a**Synopsis**

`create-pci-dec21140a [“name”] “mac_address” [“bios”]`

Description

Creates a non-instantiated component of the class “pci-dec21140a”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[pci-components](#)

create-pci-dec21140a-dml**Synopsis**

`create-pci-dec21140a-dml [“name”] “mac_address” [“bios”]`

Description

Creates a non-instantiated component of the class “pci-dec21140a-dml”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[pci-components](#)

create-pci-dec21143**Synopsis****create-pci-dec21143** [“*name*”] “*mac_address*” [“*bios*”]**Description**

Creates a non-instantiated component of the class “pci-dec21143”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By[pci-components](#)**create-pci-i21152****Synopsis****create-pci-i21152** [“*name*”]**Description**

Creates a non-instantiated component of the class “pci-i21152”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By[pci-components](#)**create-pci-i82543gc****Synopsis****create-pci-i82543gc** [“*name*”] “*mac_address*” [“*bios*”]**Description**

Creates a non-instantiated component of the class “pci-i82543gc”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By[pci-components](#)**create-pci-i82546bg****Synopsis****create-pci-i82546bg** [“*name*”] “*mac_address*” [“*bios*”]**Description**

Creates a non-instantiated component of the class “pci-i82546bg”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[pci-components](#)

create-pci-i82559

Synopsis

```
create-pci-i82559 ["name"] "mac_address" ["bios"]
```

Description

Creates a non-instantiated component of the class “pci-i82559”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[pci-components](#)

create-pci-ispl040

Synopsis

```
create-pci-ispl040 ["name"] [scsi_id] ["bios"]
```

Description

Creates a non-instantiated component of the class “pci-ispl040”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[pci-components](#)

create-pci-ispl2200

Synopsis

```
create-pci-ispl2200 ["name"] loop_id ["bios"]
```

Description

Creates a non-instantiated component of the class “pci-ispl2200”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[pci-components](#)

create-pci-pd6729

Synopsis

```
create-pci-pd6729 ["name"]
```

Description

Creates a non-instantiated component of the class “pci-pd6729”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[pci-components](#)

create-pci-pmc1553**Synopsis**

`create-pci-pmc1553 [“name”]`

Description

Creates a non-instantiated component of the class “pci-pmc1553”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[mil-std-1553-components](#)

create-pci-sil680a**Synopsis**

`create-pci-sil680a [“name”]`

Description

Creates a non-instantiated component of the class “pci-sil680a”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[pci-components](#)

create-pci-sym53c810**Synopsis**

`create-pci-sym53c810 [“name”] [“bios”]`

Description

Creates a non-instantiated component of the class “pci-sym53c810”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[pci-components](#)

create-pci-sym53c875**Synopsis**

```
create-pci-sym53c875 ["name"] ["bios"]
```

Description

Creates a non-instantiated component of the class “pci-sym53c875”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[pci-components](#)

create-pci-sym53c876**Synopsis**

```
create-pci-sym53c876 ["name"]
```

Description

Creates a non-instantiated component of the class “pci-sym53c876”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[pci-components](#)

create-pci-tsb12lv26**Synopsis**

```
create-pci-tsb12lv26 ["name"]
```

Description

Creates a non-instantiated component of the class “pci-tsb12lv26”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[firewire-components](#)

create-pci-vga**Synopsis**

```
create-pci-vga ["name"] ["bios"]
```

Description

Creates a non-instantiated component of the class “pci-vga”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[pci-components](#)

create-phy-comp

Synopsis

```
create-phy-comp ["name"] [mii_address] [phy_id]
```

Description

This command creates a non-instantiated component of the class **phy_comp**.

The class description for the **phy_comp** class: Component representing a generic IEEE 802.3 PHY

***name* is Optional**

If not specified, the component will get a class-specific default name.

***mii_address* is Optional**

PHY address on MII bus

***phy_id* is Optional**

PHY ID (i.e., vendor)

Provided By

[phy-comp](#)

create-phy-mii-transceiver

Synopsis

```
create-phy-mii-transceiver ["name"] [phy_id] [mii_address]
```

Description

Creates a non-instantiated component of the class “phy-mii-transceiver”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[phy-components](#)

create-ps2-keyboard-mouse

Synopsis

```
create-ps2-keyboard-mouse ["name"]
```

Description

Creates a non-instantiated component of the class “ps2-keyboard-mouse”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[isa-components](#)

create-rapido-link

Synopsis

create-rapido-link [“*name*”] [“*global_id*”] [*goal_latency*]

Description

This command creates a non-instantiated component of the class **rapido_link**.

The class description for the **rapido_link** class: RapidIO link connecting two RapidIO devices

***name* is Optional**

If not specified, the component will get a class-specific default name.

***global_id* is Optional**

Global identifier for use in distributed simulation or NIL if the link is not distributed.

***goal_latency* is Optional**

Goal latency in seconds for this link. See also the **set-min-latency** command.

Provided By
[rapido-link](#)

create-real-network-bridge

Synopsis

create-real-network-bridge [“*name*”] [“*interface*”] [*persistent*] [*tap_bridge*] “*access*” *no_mac_translate*

Description

Creates a non-instantiated component of the class “real-network-bridge”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[real-network](#)

create-real-network-host

Synopsis

create-real-network-host [“*name*”] “*interface*” [*persistent*]

Description

Creates a non-instantiated component of the class “real-network-host”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[real-network](#)

create-real-network-router

Synopsis

```
create-real-network-router ["name"] ["interface"] ["gateway"] "ip" "netmask"
```

Description

Creates a non-instantiated component of the class “real-network-router”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[real-network](#)

create-sample-gcache

Synopsis

```
create-sample-gcache ["name"] [assoc] [block_STC] [line_number] [line_size] ["replacement_policy"] [seed] [virtual_index] [virtual_tag] [write_allocate] [write_back] [penalty_read] [penalty_read_next] [penalty_write] [penalty_write_next]
```

Description

Creates a non-instantiated component of the class “sample-gcache”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[timing-components](#)

create-script-barrier

Synopsis

```
create-script-barrier num_branches
```

Description

Creates a script barrier that can be used for synchronization of script branches. The return value should only be used as argument to the **wait-for-script-barrier** command.

A script branch enters a barrier by calling **wait-for-script-barrier**. It will then be suspended until *num_branches* script branches have entered the barrier. Once all script branches have reached the barrier, they are released and will continue executing. At the same time the barrier is reset and can be used again.

Provided By
[Simics Core](#)

See Also

[script-branch](#), [wait-for-script-barrier](#)

create-script-pipe**Synopsis**

`create-script-pipe`

Description

Creates a script pipe that can be used to send data from one script branch, or the main branch, to another script branch. The return value should only be used as argument to the **wait-for-script-pipe** and **add-data-to-script-pipe** commands.

Provided By

[Simics Core](#)

See Also

[script-branch](#), [script-pipe-has-data](#), [wait-for-script-pipe](#), [add-data-to-script-pipe](#)

create-sdram-memory-module**Synopsis**

`create-sdram-memory-module ["name"] [rows] [columns] [ranks] [module_data_width]
[primary_width] [ecc_width] [banks] [rank_density] ["module_type"] [cas_latency]`

Description

Creates a non-instantiated component of the class “sdram-memory-module”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[memory-components](#)

create-ser-link**Synopsis**

`create-ser-link ["name"] [buffer_size] ["global_id"] [goal_latency]`

Description

This command creates a non-instantiated component of the class **ser_link**.

The class description for the **ser_link** class: Serial link connecting two serial devices

***name* is Optional**

If not specified, the component will get a class-specific default name.

***buffer_size* is Optional**

The number of characters the link may buffer. Must be at least 1.

***global_id* is Optional**

Global identifier for use in distributed simulation or NIL if the link is not distributed.

***goal_latency* is Optional**

Goal latency in seconds for this link. See also the **set-min-latency** command.

Provided By

[ser-link](#)

create-signal-link**Synopsis**

create-signal-link [“*name*”] [“*global_id*”] [*goal_latency*]

Description

This command creates a non-instantiated component of the class **signal_link**.

The class description for the **signal_link** class: The “signal_link” component represents a unidirectional signal that can be either high or low. It can be used to model electrical wires or more abstract binary signals.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***global_id* is Optional**

Global identifier for use in distributed simulation or NIL if the link is not distributed.

***goal_latency* is Optional**

Goal latency in seconds for this link. See also the **set-min-latency** command.

Provided By

[signal-link](#)

create-simple-fc-disk**Synopsis**

create-simple-fc-disk [“*name*”] *size* [“*file*”] *loop_id* *node_name* *port_name*

Description

Creates a non-instantiated component of the class “simple-fc-disk”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

create-simple-memory-module

Synopsis

```
create-simple-memory-module ["name"] [memory_megs]
```

Description

This command creates a non-instantiated component of the class **simple_memory_module**.

The class description for the **simple_memory_module** class: A simple memory module.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***memory_megs* is Optional**

MB memory.

Provided By

[memory-comp](#)

create-sio-w83627hf**Synopsis**

```
create-sio-w83627hf ["name"]
```

Description

Creates a non-instantiated component of the class "sio-w83627hf". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[isa-components](#)

create-std-etg**Synopsis**

```
create-std-etg ["name"] [“mac_address”] “ip” “netmask” “dst_ip” [“gateway_ip”] [pps]  
[packet_size] [port]
```

Description

Creates a non-instantiated component of the class "std-etg". If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

create-std-ethernet-link

Synopsis

```
create-std-ethernet-link [“name”] [latency] [“link_name”] [distributed] [frame_echo]
```

Description

Creates a non-instantiated component of the class “std-ethernet-link”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

create-std-firewire-bus**Synopsis**

```
create-std-firewire-bus [“name”]
```

Description

Creates a non-instantiated component of the class “std-firewire-bus”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[firewire-components](#)

create-std-firewire-sample-device**Synopsis**

```
create-std-firewire-sample-device [“name”]
```

Description

Creates a non-instantiated component of the class “std-firewire-sample-device”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[firewire-components](#)

create-std-generic-link**Synopsis**

```
create-std-generic-link [“name”] [latency] [“link_name”] [distributed]
```

Description

Creates a non-instantiated component of the class “std-generic-link”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[std-components](#)

create-std-generic-link-sample

Synopsis

```
create-std-generic-link-sample ["name"] address
```

Description

Creates a non-instantiated component of the class “std-generic-link-sample”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[std-components](#)

create-std-graphics-console

Synopsis

```
create-std-graphics-console ["name"] [window] ["title"]
```

Description

Creates a non-instantiated component of the class “std-graphics-console”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[std-components](#)

create-std-host-serial-console

Synopsis

```
create-std-host-serial-console ["name"] ["port"]
```

Description

Creates a non-instantiated component of the class “std-host-serial-console”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[std-components](#)

create-std-ide-cdrom

Synopsis

```
create-std-ide-cdrom ["name"]
```

Description

Creates a non-instantiated component of the class “std-ide-cdrom”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

create-std-ide-disk**Synopsis**

create-std-ide-disk [“*name*”] *size* [“*file*”]

Description

Creates a non-instantiated component of the class “std-ide-disk”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

create-std mmc-card**Synopsis**

create-std mmc-card [“*name*”] [“*file*”] *size* [“*type*”]

Description

This command creates a non-instantiated component of the class **std mmc-card**.

The class description for the **std mmc-card** class: The std mmc-card component represents an MMC/SD/SDHC/SDIO card.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***file* is Optional**

File with disk contents for the full disk. Either a raw file or a CRAFF file

***size* is Required**

Card size, in bytes

***type* is Optional**

Card type (‘mmc’, ‘sd’, ‘sdhc’ or ‘sdio’). Note that the card type will be adjusted by the model to handle large card sizes (i.e. card type will be forced to ‘sdhc’ if you create an 8 GB ‘mmc’ card).

Provided By

[std-components](#)

create-std-ms1553-link**Synopsis**

```
create-std-ms1553-link [“name”]
```

Description

Creates a non-instantiated component of the class “std-ms1553-link”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[mil-std-1553-components](#)

create-std-pcmcia-flash-disk**Synopsis**

```
create-std-pcmcia-flash-disk [“name”] size [“file”]
```

Description

Creates a non-instantiated component of the class “std-pcmcia-flash-disk”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

create-std-sata-cdrom**Synopsis**

```
create-std-sata-cdrom [“name”]
```

Description

This command creates a non-instantiated component of the class **std_sata_cdrom**.

The class description for the **std_sata_cdrom** class: The “sata_cdrom” component represents an Serial ATA CD-ROM.

***name* is Optional**

If not specified, the component will get a class-specific default name.

Provided By

[std-components](#)

create-std-sata-disk**Synopsis**

```
create-std-sata-disk [“name”] [“file”] [size]
```

Description

This command creates a non-instantiated component of the class **std_sata_disk**.

The class description for the **std_sata_disk** class: The “sata_disk” component represents a Serial ATA Disk.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***file* is Optional**

File with disk contents for the full disk. Either a raw file or a CRAFF file.

***size* is Optional**

The size of the SATA disk in bytes. If it is omitted or less than the size of the image file, the disk size will be set to the same size as the image file.

Provided By

[std-components](#)

create-std-scsi-bus**Synopsis**

create-std-scsi-bus [“*name*”]

Description

Creates a non-instantiated component of the class “std-scsi-bus”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

create-std-scsi-cdrom**Synopsis**

create-std-scsi-cdrom [“*name*”] *scsi_id*

Description

Creates a non-instantiated component of the class “std-scsi-cdrom”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

create-std-scsi-disk**Synopsis**

create-std-scsi-disk [“*name*”] *scsi_id* *size* [“*file*”]

Description

Creates a non-instantiated component of the class “std-scsi-disk”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

create-std-serial-link**Synopsis**

create-std-serial-link [“*name*”] [*latency*] [“*link_name*”] [*distributed*] [*throttle*]

Description

Creates a non-instantiated component of the class “std-serial-link”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

create-std-server-console**Synopsis**

create-std-server-console [“*name*”] [*telnet_port*]

Description

Creates a non-instantiated component of the class “std-server-console”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

create-std-service-node**Synopsis**

create-std-service-node [“*name*”]

Description

Creates a non-instantiated component of the class “std-service-node”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[service-node](#)

create-std-super-io**Synopsis**

```
create-std-super-io ["name"] [add_par_port]
```

Description

Creates a non-instantiated component of the class “std-super-io”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[isa-components](#)

create-std-telnet-console**Synopsis**

```
create-std-telnet-console ["name"] [telnet_port]
```

Description

Creates a non-instantiated component of the class “std-telnet-console”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

create-std-text-console**Synopsis**

```
create-std-text-console ["name"] [window] [title] [bg_color] [fg_color] [x11_font]  
[win32_font] [width] [height]
```

Description

Creates a non-instantiated component of the class “std-text-console”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

create-std-text-graphics-console**Synopsis**

```
create-std-text-graphics-console ["name"] [window] [title] [bg_color] [fg_color]  
[x11_font] [win32_font]
```

Description

Creates a non-instantiated component of the class “std-text-graphics-console”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

create-unconnected-ethernet-probe**Synopsis**

create-unconnected-ethernet-probe [“*name*”] [*clock*]

Description**Provided By**

[eth-probe](#)

create-usb-disk**Synopsis**

create-usb-disk [“*name*”] [*size*] [“*file*”]

Description

Creates a non-instantiated component of the class “usb-disk”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[usb-components](#)

create-usb-hs-keyboard-comp**Synopsis**

create-usb-hs-keyboard-comp [“*name*”]

Description

This command creates a non-instantiated component of the class **usb_hs_keyboard_comp**.

The class description for the **usb_hs_keyboard_comp** class: The High Speed USB Keyboard component class.

***name* is Optional**

If not specified, the component will get a class-specific default name.

Provided By
[usb-hid-components](#)

create-usb-keyboard-comp

Synopsis
create-usb-keyboard-comp [“*name*”]

Description

This command creates a non-instantiated component of the class **usb_keyboard_comp**.

The class description for the **usb_keyboard_comp** class: The USB Keyboard component class.

***name* is Optional**

If not specified, the component will get a class-specific default name.

Provided By
[usb-hid-components](#)

create-usb-mouse-comp

Synopsis
create-usb-mouse-comp [“*name*”]

Description

This command creates a non-instantiated component of the class **usb_mouse_comp**.

The class description for the **usb_mouse_comp** class: The USB Mouse component class.

***name* is Optional**

If not specified, the component will get a class-specific default name.

Provided By
[usb-hid-components](#)

create-usb-santa

Synopsis
create-usb-santa [“*name*”]

Description

Creates a non-instantiated component of the class “usb-santa”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[usb-components](#)

create-usb-tablet-comp**Synopsis**

```
create-usb-tablet-comp ["name"]
```

Description

Creates a non-instantiated component of the class “usb-tablet-comp”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[usb-components](#)

create-usb-wacom-tablet-comp**Synopsis**

```
create-usb-wacom-tablet-comp ["name"]
```

Description

Creates a non-instantiated component of the class “usb-wacom-tablet-comp”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[usb-components](#)

create-usb-xmas-tree**Synopsis**

```
create-usb-xmas-tree ["name"]
```

Description

This command creates a non-instantiated component of the class **usb_xmas_tree**.

The class description for the **usb_xmas_tree** class: The Xmas tree USB component.

***name* is Optional**

If not specified, the component will get a class-specific default name.

Provided By

[usb-comp](#)

current-namespace**Synopsis**

```
current-namespace
```

Description

Return the current namespace. This can be passed to **change-namespace** if you later want to return to the present location.

Provided By

[Simics Core](#)

See Also

[change-namespace](#)

current-processor**Synopsis**

current-processor

Description

Returns the name of the currently executing processor.

Do not use this command from scripts. It can lead to indeterminism on multi-processor targets.

Provided By

[Simics Core](#)

cycle-break**Alias**

cb

Synopsis

cycle-break [*cpu-name*] *cycles*
<cycle>.cycle-break *cycles*

Description

Sets a breakpoint so that the CPU will stop after running *cycles* number of cycles from the time the command was issued. If the CPU is not specified the selected frontend processor will be used (see **pselect**).

To list all breakpoints set use the command **list-breakpoints**.

Provided By

[Simics Core](#)

See Also

[cycle-break-absolute](#), [step-break](#), [step-break-absolute](#), [list-breakpoints](#)

cycle-break-absolute

Alias

cba

Synopsis

cycle-break-absolute [*cpu-name*] *cycles*
<cycle>.cycle-break-absolute *cycles*

Description

Set a breakpoint so that the selected CPU will stop after its cycle counter has reached the *cycles* value. If the CPU is not specified the selected frontend processor will be used (see **pselect**).

To list all breakpoints set use the command **list-breakpoints**.

Provided By[Simics Core](#)**See Also**[cycle-break](#), [step-break](#), [step-break-absolute](#), [list-breakpoints](#)**date****Synopsis****date** [-t]**Description**

Return the current date and time on the host, in the form Fri Nov 7 12:00:36 2008. If the -t flag used, the time is returned in seconds as a floating point value.

Provided By[Simics Core](#)**debug-context****Synopsis****debug-context** [*object*|“*context-query*”] [“*name*”]**Description**

Optionally change the current debug context and return it. The current debug context can be specified either using a memory-space or processor in Simics, or a debug context specified either through as a debug context object or a context query.

The current debug context is the debug context global stepping and debugger inspection commands will affect.

Provided By[tcf-agent](#)**dec**

Synopsis**dec** *value* [-u]**Description**

Returns the parameter as a string in decimal notation. This is similar to **print -d** *value*. To ignore any default digit grouping, the **-u** (unformatted) flag is used.

Provided By[Simics Core](#)**See Also**[print](#), [hex](#), [bin](#), [oct](#), [digit-grouping](#)**default-port-forward-target****Synopsis****default-port-forward-target** [“*target-ip*”]**Description**

Sets the IP address of a simulated machine that will be used by the **connect-real-network** command if none is given as argument. This is useful in single machine configurations where the same IP is used all the time.

Provided By[service-node](#)**See Also**[connect-real-network](#), [connect-real-network-port-in](#)**defined****Synopsis****defined** “*variable*”**Description**

Returns TRUE if *variable* is a defined CLI variable, and FALSE if not.

Note that **defined** *foo* tests whether the variable *foo* is defined, whereas **defined** *\$bar* tests whether the variable whose name is stored in *\$bar* is defined.

Provided By[Simics Core](#)**See Also**[read-variable](#), [\\$](#), [list-variables](#)**delete**

Synopsis

```
delete [ id ... ] [-all]
```

Description

Removes the breakpoints matching the ID numbers *id*.

Use **list-breakpoints** to list all breakpoints' ID.

If **-all** is given, delete all breakpoints.

Provided By

[Simics Core](#)

See Also

[`<breakpoint>.break`](#), [`enable`](#), [`ignore`](#), [`set-prefix`](#), [`set-substr`](#), [`set-pattern`](#), [`list-breakpoints`](#)

delete-bookmark**Synopsis**

```
delete-bookmark ("bookmark"|-all)
```

Description

Deletes a time bookmark. Deleting all time bookmarks can improve forward simulation performance.

Reverse operations are possible in the region following the first (i.e. oldest) time bookmark.

Provided By

[Simics Core](#)

See Also

[`set-bookmark`](#), [`list-bookmarks`](#), [`reverse`](#)

devs**Synopsis**

```
devs [object-name]
```

Description

Print a list of all devices in Simics, with information about how many times each device has been accessed, and where they are mapped.

The mappings are presented as start and end offsets within a named memory space.

The function number or port name associated with each different mapping for a device is also printed.

Provided By

[Simics Core](#)

digit-grouping**Synopsis****digit-grouping** *base* [*digits*]**Description**

Changes or displays how numbers are formatted for a given radix. This will separate groups of *digits* digits by an underscore when they are formatted for output. Separate grouping is maintained for each radix. If *digits* is zero, no separators are printed for that radix.

Provided By[Simics Core](#)**See Also**[output-radix](#), [print](#), [hex](#), [dec](#), [oct](#), [bin](#)**dirs****Synopsis****dirs****Description**

Shows the contents of the directory stack.

Provided By[Simics Core](#)**See Also**[pushd](#), [popd](#)**disable****Synopsis****disable** (-all|*id*)**enable** (-all|*id*)**Description**

Enable/disable instruction breakpoint with ID number *id*, or all breakpoints if **-all** is specified.

Use **list-breakpoints** list breakpoint IDs.

While Simics will not stop on a disabled breakpoint, it will still be counted.

Provided By[Simics Core](#)**See Also**[`<breakpoint>.break`](#), [`delete`](#), [`ignore`](#), [`list-breakpoints`](#)

disable-hypersim**Synopsis**

```
disable-hypersim [-v]
enable-hypersim [-v]
```

Description

Enables or disables hyper-simulation. By enabling hyper-simulation all available hypersim classes are connected to all available processors. Hypersim modules are target and OS specific -- different modules are included in various add-on packages. Exactly which hypersim modules are available is a function of which add-on packages you have installed. Enable verbose mode with the flag *-v*. The names of all hypersim-pattern-matcher objects used are returned.

Provided By

[hypersim-pattern-matcher](#)

disable-magic-breakpoint**Synopsis**

```
disable-magic-breakpoint
enable-magic-breakpoint
magic-breakpoint-enabled
```

Description

The **enable-magic-breakpoint** command sets a breakpoint on the magic instruction with argument 0, except on SPARC targets. On SPARC targets it is a magic instruction with the top 6 bits of the 22-bit parameter field set to binary 000100. The breakpoint can be removed with the **disable-magic-breakpoint** command. To break on all magic instructions, use `break-hap Core_Magic_Instruction`.

Provided By

[Simics Core](#)

disable-mtprof**Synopsis**

```
disable-mtprof
enable-mtprof [interval]
```

Description

Enable multithreaded simulation profiling. The amount of host cpu time required to simulate each cell is measured every *interval* virtual ms.

The collected data is fed into a performance model which estimates how fast the simulation would run on a system with enough host cores and Simics Accelerator licenses to allow each cell to run on a dedicated core. The performance model also

gives some insights into the performance implications of various min-latency settings (settable though the **set-min-latency** commands).

The *interval* parameter should normally be set to a value of the same order as the min-latency setting of **intereset**.

Provided By

[mtprof](#)

See Also

[`<mtprof>.cellstat`](#), [`<mtprof>.modelstat`](#), [`<mtprof>.save-data`](#)

disable-multithreading

Synopsis

disable-multithreading
enable-multithreading [-force]

Description

Enable or disable the use of multiple concurrent threads for running the simulation. Running with multiple threads requires that all loaded modules are thread safe, and that the objects are correctly partitioned into cells. The use of multiple threads can be forced on, even when unsafe modules are loaded or when the cell partitioning rules are violated, by using the *-force* flag. The *-force* flag should only be used during model development.

Provided By

[Simics Core](#)

See Also

[`set-thread-limit`](#)

disable-page-sharing

Synopsis

disable-page-sharing
enable-page-sharing [-now]

Description

Enable or disable the page-sharing feature, which detects identical pages in the systems being simulated. Pages can be RAM, ROM, flash, disk, or any page represented by **image** objects.

Pages detected to have identical contents are replaced with a single read-only copy, consequently reducing the amount of host memory used. If an identical page is written to, it will again get a private writable instance of the page.

Identical pages are detected during run-time by certain triggers. If the *-now* switch is used, all currently active pages will be examined for sharing directly. Disabling

page-sharing only means that no more pages will be shared, pages which have already been shared will not be un-shared.

Provided By
[Simics Core](#)

disable-real-time-mode

Synopsis

```
disable-real-time-mode
<realtime>.disable
<realtime>.enable [speed] [check-interval] [drift-compensate]
enable-real-time-mode [speed] [check_interval] [drift_compensate]
```

Description

In some cases simulated time may run faster than real time; this can happen if the OS is in a tight idle loop or an instruction halts execution waiting for an interrupt, or if the host machine is simply sufficiently fast. This can cause problems for programs that interact with the real world (for example the user), since time-outs may expire really fast.

A **realtime** object will, when enabled, periodically check the simulation speed and wait for a while if it is too high. *speed* specifies how fast simulated time is allowed to run, in percent of real time; default is 100. *check-interval* specifies how often the check should take place, in milliseconds of simulated time; default is 1000. Higher values give better performance; lower values reduce the maximum difference between real and simulated time. Setting this to less than **cpu-switch-time** has no effect.

The *speed* argument says how fast the simulation *should* run, but the actual speed will always deviate a little from that value even if the host is fast enough. To keep these errors from accumulating, the simulation speed has to be adjusted; *drift-compensate* regulates how much it may be adjusted. If set to (for example) 0.25, simulation speed may be increased or decreased by up to 25% if necessary to make up for any accumulated drift with respect to real time. If set to zero (the default), the simulation speed may not be changed at all from its set value.

enable and **disable** work on a given **realtime** object; **enable-real-time-mode** and **disable-real-time-mode** work on a default **realtime** object.

Provided By
[realtime](#)

disable-reverse-execution

Synopsis

```
disable-reverse-execution
```

Description

Deletes all time bookmark and disables reverse execution. This can improve forward simulation performance.

Provided By[Simics Core](#)**See Also**[enable-reverse-execution](#), [set-bookmark](#), [list-bookmarks](#), [reverse](#)**disassemble****Alias**

da

Synopsis**disassemble** [*cpu-name*] [*address*] [*count*] [*bytes*]**Description**

Disassembles *count* instructions or *bytes* octets starting at *address* for processor *cpu-name*. If the processor is not given the current frontend processor will be used. The method variant can also be used to select a processor; e.g., **cpu0.disassemble**.

On some architectures, *address* must be word aligned. A physical address is given by prefixing the address with **p**: (e.g., **p:0xf000**). With no prefix, a virtual address will be assumed. If the address is omitted the current program counter will be used. *count* defaults to 1 instruction.

Global disassembly settings, such as whether to print the raw opcodes, can be set by the **disassemble-settings** command.

If supported by the processor, this command will also include various profiling statistics for the address of each instruction. One column is printed for each selected profiler. See the **aprof-views** command in each processor for more information.

Provided By[Simics Core](#)**See Also**[x](#), [disassemble-settings](#)**disassemble-settings****Synopsis****disassemble-settings** [*opcode*] [*physaddr*] [*partial-opcode*] [*turbo*]**Description**

Change disassemble output settings. Each of these settings can be set to **on** or **off**.

opcode indicates whether to print the raw bytes of the instruction in addition to the disassembly. If *partial-opcode* is set, and the opcode encodes more than one instruction (which can be the case on VLIW architectures), the opcode bytes will be divided among the instructions so that the entire opcode has been printed exactly once when all the

instructions have been disassembled. If *partial-opcode* is not set, the entire opcode will be printed for every instruction.

physaddr indicates whether to compute and display the physical address if the virtual address was specified (if the physical address was specified, the virtual address is never printed).

Without arguments, the current settings will be shown.

Provided By

[Simics Core](#)

See Also

[disassemble](#)

disconnect

Synopsis

disconnect [*cnt0*] [*cnt1*]

Description

Disconnect connector *cnt0* from *cnt1*.

Provided By

[Simics Core](#)

disconnect-real-network

Synopsis

```
disconnect-real-network
<ethernet-link>.disconnect-real-network
<ethernet_cable>.disconnect-real-network
<ethernet_hub>.disconnect-real-network
<ethernet_switch>.disconnect-real-network
```

Description

Closes all connections to real networks except port forwarding and NAPT.

Provided By

[real-network](#)

See Also

[connect-real-network-host](#), [connect-real-network-bridge](#)

disconnect-real-network-port-in

Synopsis

disconnect-real-network-port-in (*target-port|“service”*) *ethernet-link* [*service-node*] [“*host-ip*”] [*host-port*] [“*target-ip*”] [-tcp] [-udp]

connect-real-network-port-in (*target-port|“service”*) *ethernet-link* [*service-node*] [“*host-ip*”] [*host-port*] [“*target-ip*”] [-tcp] [-udp] [-f]

Description

Enables or disables port forwarding from the host that Simics is running on, to a simulated machine, specified by *target-ip*. The externally visible port *host-port* on the host is mapped to the port *target-port* on the simulated machine. For commonly used services the string argument *service* can be used instead of a port number. If several Ethernet links exists, the one that the simulated machine is connected to must be specified.

The flags *-tcp* and *-udp* can be used to specify the protocol to forward. The default is to forward only the usual protocol for named services and both *tcp* and *udp* for numerically specified ports.

The flag *-f* can be used to cause the command to fail if the suggested host port could not be allocated, without the flag the command will assign the first available port starting from the specified host port and upwards.

The *host-port* given is only a hint, and the actual port used may be a different one. The command output shows the actual port used, and it can also be determined by inspecting the connections attribute in the appropriate port forwarding object.

Provided By

[service-node](#)

See Also

[connect-real-network](#), [connect-real-network-port-out](#), [connect-real-network-napt](#), [connect-real-network-host](#), [connect-real-network-bridge](#)

disconnect-real-network-port-out**Synopsis**

disconnect-real-network-port-out *service-node-port* *ethernet-link* [*service-node*] “*target-ip*” *target-port* [-tcp] [-udp]

connect-real-network-port-out *service-node-port* *ethernet-link* [*service-node*] “*target-ip*” *target-port* [-tcp] [-udp]

Description

Enables port forwarding to a machine on the real network. Traffic targeting port *service-node-port* on the service node connected to *ethernet-link* will be forwarded to port *target-port* on *target-ip*.

Both *tcp* and *udp* will be forwarded unless one of the *-tcp* or *-udp* flags are given in which case only that protocol will be forwarded.

Provided By
[service-node](#)

See Also

[connect-real-network](#), [connect-real-network-port-in](#), [connect-real-network-napt](#), [connect-real-network-host](#), [connect-real-network-bridge](#)

display

Synopsis

`display ["expression"] [-l] [-p] [-t]`

Description

Install a Python expression, or a frontend statement that will be printed when Simics returns to the prompt. The `-p` flag is used to indicate that the string argument is in Python. To list all installed display expressions, the `-l` argument should be used. The expressions are only evaluated and printed if the simulation has run any instructions since last time, but a re-evaluation can be forced by calling `display` with no arguments. The `-t` argument makes the output be tagged in a way that makes it possible to capture the output by external means.

Provided By
[Simics Core](#)

See Also

[undisplay](#)

down

Synopsis

`down [N]`

Description

Moves `N` frames down the stack (towards the innermost frame). `N` defaults to one.

Provided By
[Simics Core](#)

See Also

[frame](#), [up](#), [stack-trace](#)

dstc-disable

Synopsis

`dstc-disable`
`dstc-enable`
`iostc-disable`

iostc-enable
istc-disable
istc-enable
stc-status

Description

These commands are for advanced usage only. They allow the user to control the usage of Simics-internal caches.

The Simulator Translation Caches (STCs) are designed to increase execution performance. The D-STC caches data translations (logical to physical to real (host) address). The I-STC caches instruction translations of taken jumps. Finally, the IO-STC caches logical to physical address translation for device accesses.

By default the STCs are **on**.

When a memory hierarchy is connected (such as a cache module) it must have been designed to work along with the STCs, or it may not be called for all the memory transactions it is interested in. These commands can be used to detect if too many translations are kept in the STCs, causing the simulation to be incorrect.

Turning the STCs off means that current contents will be flushed and no more entries will be inserted into the STCs.

Provided By

[Simics Core](#)

dstc-enable

Synopsis

dstc-enable
dstc-disable
iostc-disable
iostc-enable
istc-disable
istc-enable
stc-status

Description

These commands are for advanced usage only. They allow the user to control the usage of Simics-internal caches.

The Simulator Translation Caches (STCs) are designed to increase execution performance. The D-STC caches data translations (logical to physical to real (host) address). The I-STC caches instruction translations of taken jumps. Finally, the IO-STC caches logical to physical address translation for device accesses.

By default the STCs are **on**.

When a memory hierarchy is connected (such as a cache module) it must have been designed to work along with the STCs, or it may not be called for all the memory

transactions it is interested in. These commands can be used to detect if too many translations are kept in the STCs, causing the simulation to be incorrect.

Turning the STCs off means that current contents will be flushed and no more entries will be inserted into the STCs.

Provided By

[Simics Core](#)

echo

Synopsis

echo [*integer|float|"string"|list*]

Description

Prints the string, integer, or float. Useful for annotating test scripts.

Provided By

[Simics Core](#)

else

Synopsis

```
if condition { commands }
if condition { commands } else { commands }
if condition { commands } else if condition { commands }
```

Description

Runs a block of commands conditionally.

The **if** command returns the value of the last executed command in the block.

Provided By

[Simics Core](#)

See Also

[while](#)

enable

Synopsis

```
enable (-all|id)
disable (-all|id)
```

Description

Enable/disable instruction breakpoint with ID number *id*, or all breakpoints if **-all** is specified.

Use **list-breakpoints** list breakpoint IDs.

While Simics will not stop on a disabled breakpoint, it will still be counted.

Provided By[Simics Core](#)**See Also**[`<breakpoint>.break, delete, ignore, list-breakpoints`](#)**`enable-core2-bugfix`****Synopsis****`enable-core2-bugfix`****Description**

Enable workaround for an Intel® Core™2 hardware bug.

Provided By[trace-sync](#)**`enable-hypersim`****Synopsis****`enable-hypersim [-v]`**
`disable-hypersim [-v]`**Description**

Enables or disables hyper-simulation. By enabling hyper-simulation all available hypersim classes are connected to all available processors. Hypersim modules are target and OS specific -- different modules are included in various add-on packages. Exactly which hypersim modules are available is a function of which add-on packages you have installed. Enable verbose mode with the flag *-v*. The names of all hypersim-pattern-matcher objects used are returned.

Provided By[hypersim-pattern-matcher](#)**`enable-magic-breakpoint`****Synopsis****`enable-magic-breakpoint`**
`disable-magic-breakpoint`
`magic-breakpoint-enabled`**Description**

The **enable-magic-breakpoint** command sets a breakpoint on the magic instruction with argument 0, except on SPARC targets. On SPARC targets it is a magic instruction with the top 6 bits of the 22-bit parameter field set to binary 000100. The breakpoint can be removed with the **disable-magic-breakpoint** command. To break on all magic instructions, use `break-hap Core_Magic_Instruction`.

Provided By
[Simics Core](#)

enable-mtprof

Synopsis

enable-mtprof [*interval*]
disable-mtprof

Description

Enable multithreaded simulation profiling. The amount of host cpu time required to simulate each cell is measured every *interval* virtual ms.

The collected data is fed into a performance model which estimates how fast the simulation would run on a system with enough host cores and Simics Accelerator licenses to allow each cell to run on a dedicated core. The performance model also gives some insights into the performance implications of various min-latency settings (settable through the **set-min-latency** commands).

The *interval* parameter should normally be set to a value of the same order as the min-latency setting of interest.

Provided By
[mtprof](#)

See Also

[`<mtprof>.cellstat`](#), [`<mtprof>.modelstat`](#), [`<mtprof>.save-data`](#)

enable-multithreading

Synopsis

enable-multithreading [-force]
disable-multithreading

Description

Enable or disable the use of multiple concurrent threads for running the simulation. Running with multiple threads requires that all loaded modules are thread safe, and that the objects are correctly partitioned into cells. The use of multiple threads can be forced on, even when unsafe modules are loaded or when the cell partitioning rules are violated, by using the *-force* flag. The *-force* flag should only be used during model development.

Provided By
[Simics Core](#)

See Also

[`set-thread-limit`](#)

enable-page-sharing**Synopsis**

```
enable-page-sharing [-now]
disable-page-sharing
```

Description

Enable or disable the page-sharing feature, which detects identical pages in the systems being simulated. Pages can be RAM, ROM, flash, disk, or any page represented by **image** objects.

Pages detected to have identical contents are replaced with a single read-only copy, consequently reducing the amount of host memory used. If an identical page is written to, it will again get a private writable instance of the page.

Identical pages are detected during run-time by certain triggers. If the *-now* switch is used, all currently active pages will be examined for sharing directly. Disabling page-sharing only means that no more pages will be shared, pages which have already been shared will not be un-shared.

Provided By

[Simics Core](#)

enable-real-time-mode**Synopsis**

```
enable-real-time-mode [speed] [check_interval] [drift-compensate]
<realtime>.disable
<realtime>.enable [speed] [check-interval] [drift-compensate]
disable-real-time-mode
```

Description

In some cases simulated time may run faster than real time; this can happen if the OS is in a tight idle loop or an instruction halts execution waiting for an interrupt, or if the host machine is simply sufficiently fast. This can cause problems for programs that interact with the real world (for example the user), since time-outs may expire really fast.

A **realtime** object will, when enabled, periodically check the simulation speed and wait for a while if it is too high. *speed* specifies how fast simulated time is allowed to run, in percent of real time; default is 100. *check-interval* specifies how often the check should take place, in milliseconds of simulated time; default is 1000. Higher values give better performance; lower values reduce the maximum difference between real and simulated time. Setting this to less than **cpu-switch-time** has no effect.

The *speed* argument says how fast the simulation *should* run, but the actual speed will always deviate a little from that value even if the host is fast enough. To keep these errors from accumulating, the simulation speed has to be adjusted; *drift-compensate* regulates how much it may be adjusted. If set to (for example) 0.25, simulation speed may be increased or decreased by up to 25% if necessary to make up for any accumulated

drift with respect to real time. If set to zero (the default), the simulation speed may not be changed at all from its set value.

enable and **disable** work on a given **realtime** object; **enable-real-time-mode** and **disable-real-time-mode** work on a default **realtime** object.

Provided By
[realtime](#)

enable-reverse-execution

Synopsis
enable-reverse-execution

Description

Enable reverse exection.

Reverse operations are possible in the region following the first (i.e. oldest) time bookmark.

Setting a time bookmark can cause a certain reduction in forward simulation performance (deleting all bookmarks will restore the original performance).

Provided By
[Simics Core](#)

See Also

[disable-reverse-execution](#), [set-bookmark](#), [delete-bookmark](#), [reverse](#), [skip-to](#)

env

Synopsis
env [-x] "variable"

Description

Returns the value of an environment variable. The **-x** flag can be used to test if a named variable exists.

Provided By
[Simics Core](#)

except

Synopsis
get-error-command
get-error-file
get-error-line
get-error-message
try { commands } except { on-error-commands }

Description

Runs all *commands* from the **try** part until an error occurs. If no error is encountered, *on-error-commands* are ignored, but if an error does occur the *on-error-commands* are run. Information about the error is available in the **except** part through the **get-error-commands**, **get-error-message**, **get-error-file** and **get-error-line** commands.

Provided By

[Simics Core](#)

exec**Synopsis**

exec "cmd"

Description

Executes the *cmd* argument as a CLI command line and returns the return value if any.

Provided By

[Simics Core](#)

exec-info**Synopsis**

exec-info

Description

Print CPU engine execution statistics.

Provided By

[symtable](#)

exec-info-clean**Synopsis**

exec-info-clean

Description

Reset CPU engine execution statistics. Also resets turbo stats.

Provided By

[symtable](#)

expect**Synopsis**

expect arg1 arg2 [-v]

Description

If values *arg1* and *arg2* are not equal the simulator will print them and exit with error *exit(1)*. *-v* prints the two values before comparing them.

This can be useful when writing scripts that want to assert a state in the simulator. Note that it only works with integer and string arguments.

Provided By

[Simics Core](#)

file-exists**Synopsis**

file-exists “*filename*”

Description

Looks for the file or directory *filename* in the Simics search path, starting with the current working directory. Returns true if the file is found and false if not.

Provided By

[Simics Core](#)

See Also

[resolve-file](#)

finish-function**Alias**

finish, *fin*

Synopsis

finish-function

Description

finish-function causes the simulation to run until the current function has returned.

Provided By

[Simics Core](#)

See Also

[next-instruction](#), [next-line](#), [reverse-next-instruction](#), [reverse-next-line](#), [reverse-step-instruction](#), [reverse-step-line](#), [step-instruction](#), [step-line](#), [uncall-function](#)

foreach**Synopsis**

foreach \$*iterator* in *list* { *commands* }

Description

Runs a block of commands with the variable `$iterator` set to each of the entries in `list`. The `$iterator` variable is only defined within the command block.

Provided By

[Simics Core](#)

See Also

[while](#)

frame**Alias**

f

Synopsis

frame [*frame-number*]

Description

Changes current stack frame to *frame-number*, or displays the current frame.

Provided By

[Simics Core](#)

See Also

[stack-trace](#), [psym](#), [up](#), [down](#)

function-profile**Synopsis**

function-profile *profiler* [*view*] [*maxfuncs*] [*cutoff*]

Description

Lists the most interesting functions by their number of profile counts, as determined by *profiler* (by default, view 0 of the profiler is used; this can be changed with the *view* parameter). No more than *maxfuncs* functions (10 by default) are listed, and not functions with fewer hits than *cutoff*.

Provided By

[symtable](#)

get**Synopsis**

```
get address [size] [-l] [-b]
<memory-space>.get address [size] [-l] [-b]
<memory-space>.read address [size] [-l] [-b]
<port-space>.get address [size] [-l] [-b]
<port-space>.read address [size] [-l] [-b]
```

Description

Get value of physical memory location. The size argument specifies how many bytes should be read. This defaults to 4, but can be any number of bytes between 1 and 8 (inclusive) for memory-spaces and between 1 and 8 (inclusive) for port-spaces.

The **-l** and **-b** flags are used to select little-endian and big-endian byte order, respectively. If neither is given, the byte order of the currently selected processor is used.

The **get** command performs the access in inquiry mode without triggering any side-effects while **read** may trigger side-effects.

The non-namespace version of this command operates on the physical memory associated with the current processor.

Provided By

[Simics Core](#)

See Also

[x](#), [set](#), [signed](#)

get-breakpoint-list**Synopsis**

get-breakpoint-list [-all]
list-breakpoints [-all]

Description

Prints information about all breakpoints set. The following information is printed for memory breakpoints: the id (used by other commands to refer to the breakpoint), if the breakpoint is set on physical or virtual addresses and the access type (r = read, w = write, or x = execute), if enabled (see the **enable** command), the address range of the breakpoint, how many times the breakpoint has been triggered, and what memory space or context object it is set in.

If prefix, substring and/or pattern conditions are set it will be printed as well (see **set-prefix**, **set-substr** and **set-pattern** command).

Time breakpoints are also listed.

If **-all** is passed as argument, **list-breakpoints** will also list all internal breakpoints set for simulation purposes.

The **get-breakpoint-list** return a list of all breakpoint numbers instead of printing information about the breakpoints.

Provided By

[Simics Core](#)

See Also

[`<breakpoint>.break`](#), [`delete`](#), [`enable`](#), [`ignore`](#), [`set-prefix`](#), [`set-substr`](#), [`set-pattern`](#)

get-class-list**Synopsis**

```
get-class-list [-l]
list-classes [-l]
```

Description

Print (**list-classes**) or return (**get-class-list**) a list of all available configuration classes. The **-l** flag will reduce the list to classes that has been registered by loaded modules.

Provided By

[Simics Core](#)

get-component-list**Synopsis**

```
get-component-list [component] ["class"] [-t] [-all] [-recursive]
list-components [component] ["class"] [-s] [-v] [-t] [-all] [-recursive]
```

Description

Lists components with their names, types, and connectors; and for each connector, the destination component and connector. With the *class* argument, you can restrict the listing to components of a particular class.

By default, only the components in the current namespace are listed (see the **change-namespace** command). If no current namespace has been selected, only top-level components are shown. The *component* argument can be used to override the current namespace.

With the **-all** flag, list all components, regardless of where they live. With the **-recursive** flag, in addition to listing all components that are in the selected namespace, also list all components in the namespaces below it.

Note: If the *legacy_conf* preferences option is set to true, all components belong to the top level, so most of these options have no effect.

The **-s** and **-v** flags select a briefer listing (the default), and a more verbose listing, respectively. With **-t**, list only top-level components.

The **get-component-list** command is similar to **list-components**, but returns a list of components instead of printing them. It does not recognize the **-s** and **-v** flags.

Provided By

[Simics Core](#)

get-component-prefix — deprecated**Synopsis**

```
get-component-prefix
```

Description

This command is deprecated; use [hierarchical components](#) instead.

Get the current component name prefix.

Provided By

[Simics Core](#)

See Also

[set-component-prefix](#)

get-error-command**Synopsis**

```
get-error-command
get-error-file
get-error-line
get-error-message
try { commands } except { on-error-commands }
```

Description

Runs all *commands* from the **try** part until an error occurs. If no error is encountered, *on-error-commands* are ignored, but if an error does occur the *on-error-commands* are run. Information about the error is available in the **except** part through the **get-error-commands**, **get-error-message**, **get-error-file** and **get-error-line** commands.

Provided By

[Simics Core](#)

get-error-file**Synopsis**

```
get-error-file
get-error-command
get-error-line
get-error-message
try { commands } except { on-error-commands }
```

Description

Runs all *commands* from the **try** part until an error occurs. If no error is encountered, *on-error-commands* are ignored, but if an error does occur the *on-error-commands* are run. Information about the error is available in the **except** part through the **get-error-commands**, **get-error-message**, **get-error-file** and **get-error-line** commands.

Provided By

[Simics Core](#)

get-error-line**Synopsis**

```
get-error-line
get-error-command
get-error-file
get-error-message
try { commands } except { on-error-commands }
```

Description

Runs all *commands* from the **try** part until an error occurs. If no error is encountered, *on-error-commands* are ignored, but if an error does occur the *on-error-commands* are run. Information about the error is available in the **except** part through the **get-error-commands**, **get-error-message**, **get-error-file** and **get-error-line** commands.

Provided By

[Simics Core](#)

get-error-message**Synopsis**

```
get-error-message
get-error-command
get-error-file
get-error-line
try { commands } except { on-error-commands }
```

Description

Runs all *commands* from the **try** part until an error occurs. If no error is encountered, *on-error-commands* are ignored, but if an error does occur the *on-error-commands* are run. Information about the error is available in the **except** part through the **get-error-commands**, **get-error-message**, **get-error-file** and **get-error-line** commands.

Provided By

[Simics Core](#)

get-object-list**Synopsis**

```
get-object-list ["type"] [component] [-all] [-recursive]
list-objects ["type"] [component] [-n] [-a] [-all] [-recursive]
```

Description

Lists configuration objects and the class they belong to. With the *type* argument, you can restrict the listing to objects of a particular class, or to objects that implement a particular interface.

By default, only the objects in the current namespace are listed (see the **change-namespace** command). The *component* argument can be used to override the current namespace.

Use the **-recursive** flag to list objects in both the current namespace and in the namespaces below it in the hierarchy.

With the **-all** flag, all objects are listed, regardless of namespace.

The objects are sorted by class name by default. The **-n** flag sorts them by name instead. Component objects are printed first and then all other objects. To mix all objects in the same list, use the **-a** flag.

The **get-object-list** command is similar to **list-objects**, but returns a list of object names instead of printing them. It does not recognize the **-n** and **-a** flags.

Provided By

[Simics Core](#)

See Also

[list-namespaces](#), [change-namespace](#)

help

Alias

h, man

Synopsis

help [“*topic*”]

Description

Prints help information on *topic*. *topic* can be a command, a class, an object, an interface, a module, a hap, an attribute or a function or type from the Simics API. Here are some usage examples of the **help** command:

```
simics> help ptime
[... ptime command documentation ...]

simics> help cpu0.disassemble
[... <processor>.disassemble command documentation ...]

simics> help <processor>.disassemble
[... <processor>.disassemble command documentation ...]

simics> help cpu0
[... <ppc440gp> class documentation ...]

simics> help ppc440gp
[... <ppc440gp> class documentation ...]
```

```

simics> help processor
[... <processor> interface documentation ...]

simics> help cpu0.freq_mhz
[... <ppc440gp>.freq_mhz attribute documentation ...]

simics> help ppc440gp.msr
[... <ppc440gp>.msr attribute documentation ...]

simics> help Core_Exception
[... Core_Exception hap documentation ...]

simics> help SIM_get_mem_op_type
[... SIM_get_mem_op_type() function declaration ...]

```

To refine your search and improve the tab-completion choices, you may use filters in the topic as shown below:

help topic = command:break

The recognized filters are command:, class:, object:, interface:, module:, hap:, attribute: and api:.

By default, the help command does not provided tab-completion on *topic* for modules and api symbols unless the specific filter is provided.

Provided By

[Simics Core](#)

See Also

[help-search](#), [api-help](#), [api-search](#), [win-help](#)

help-search

Alias

a, apropos, search

Synopsis

help-search [-r] “*string*”

Description

Use **help-search** *string* to list all commands for which the documentation contains the text *string*. If the *-r* flag is used, interpret *string* as a regular expression.

This command will only display search results for attributes, commands, classes, haps, and interfaces.

Provided By

[Simics Core](#)

See Also[api-search](#), [help](#), [api-help](#)**hex****Synopsis****hex** *value* [-u] [-p]**Description**

Returns the parameter as a string in hexadecimal notation. This is similar to **print -x** *value*. To ignore any default digit grouping, the **-u** (unformatted) flag is used, while **-p** removes the radix prefix **0x**.

Provided By[Simics Core](#)**See Also**[print](#), [bin](#), [oct](#), [dec](#), [digit-grouping](#)**hypersim-status****Synopsis****hypersim-status** [-v]**Description**

Show status overview for hyper-simulation. With the **-v** all <hypersim-pattern-matcher>.status commands are executed also with more detailed information.

Provided By[hypersim-pattern-matcher](#)**if****Synopsis**

```
if condition { commands }
if condition { commands } else { commands }
if condition { commands } else if condition { commands }
```

Description

Runs a block of commands conditionally.

The **if** command returns the value of the last executed command in the block.

Provided By[Simics Core](#)**See Also**[while](#)

ignore**Synopsis****ignore** *id num***Description**

Sets the ignore count for a breakpoint. This means that the next *num* times the breakpoint is reached it will not trigger (hap handlers will not be called). To break next time set *num* to 0.

Provided By[Simics Core](#)**See Also**[enable](#), [list-breakpoints](#)**in****Synopsis****needle in haystack****Description**

Returns true if the argument *needle* can be found in *haystack* and false if not. The first argument is an integer, string, floating point or boolean value and the second is a string or list. When searching in a string, the first argument must also be a string.

Provided By[Simics Core](#)**in-list****Synopsis****in-list needle haystack****Description**

Returns true if the argument *needle* can be found in *haystack* and false if not. The first argument is an integer, string, floating point or boolean value and the second is a list.

Provided By[Simics Core](#)**insert-ethernet-probe****Synopsis****insert-ethernet-probe** [“*name*”] *device* [“*attribute*”]**Description**

Provided By
[eth-probe](#)

instantiate-components

Synopsis

instantiate-components [-v] [component ...]

Description

Instantiates non-instantiated components. With no argument, all top-level components are collected and instantiated along with all components below them. If one or more components are specified as arguments, only these and the components below them are instantiated.

The *-v* flag will make the command print the name of the instantiated components. Instantiating components will discard any reverse execution history.

Provided By
[Simics Core](#)

instruction-fetch-mode

Alias

ifm

Synopsis

instruction-fetch-mode ["mode"]

Description

This command selects how instruction fetches are sent for the memory hierarchy during simulation. If set to *no-instruction-fetch*, the memory hierarchy will not receive any instruction fetch. If set to *instruction-cache-access-trace*, the memory hierarchy will receive exactly one instruction fetch every time a new cache line is accessed. The size of this cache line is defined by the attribute *instruction-fetch-line-size* in the processor object. If set to *instruction-fetch-trace*, all instruction fetches will be visible.

Note that for x86 CPUs, *instruction-cache-trace-access* is not available. On some other CPU architectures, *instruction-fetch-trace* is actually *instruction-cache-trace-access* with a line size equal to the instruction size (SPARC-V9).

Using this command without argument displays the current mode.

Provided By
[Simics Core](#)

int-to-double-float

Synopsis

int-to-double-float value

Description

Returns the parameter when interpreting the parameter as a IEEE 64-bit floating point number.

Provided By

[Simics Core](#)

int-to-extended-double-float**Synopsis**

int-to-extended-double-float *value*

Description

Returns the parameter when interpreting the parameter as a x87 80-bit floating point number.

Provided By

[Simics Core](#)

int-to-quad-float**Synopsis**

int-to-quad-float *value*

Description

Returns the parameter when interpreting the parameter as a IEEE 128-bit floating point number.

Provided By

[Simics Core](#)

int-to-single-float**Synopsis**

int-to-single-float *value*

Description

Returns the parameter when interpreting the parameter as a IEEE 32-bit floating point number.

Provided By

[Simics Core](#)

interrupt-script**Synopsis**

interrupt-script [“*message*”] [-error]

Description

Interrupts the execution of a script and prints out *message*. If *message* is not specified, a generic message is printed.

The `-error` flag tells Simics to consider the interruption as an error.

Provided By

[Simics Core](#)

See Also

[try](#)

interrupt-script-branch**Synopsis**

interrupt-script-branch *id*

Description

Send an interrupt exception to a script branch. The argument is the script branch ID, as returned by the **script-branch** command. The ID is also listed by the **list-script-branches** command. The branch will immediately wake up from any pending waits and exit when it receives the exception.

Provided By

[Simics Core](#)

See Also

[script-branch](#), [list-script-branches](#)

io-stats**Synopsis**

io-stats [*cutoff*]

Description

Gives an overview of how many devices accesses that has happened during the simulation and provides top lists for the most frequently accessed device classes and objects. The *cutoff* is default 1.0%, which means that classes or objects with lower than 1.0% of the total number of io accesses will be discarded.

Provided By

[Simics Core](#)

See Also

[devs](#)

iostc-disable

Synopsis

iostc-disable
dstc-disable
dstc-enable
iostc-enable
istc-disable
istc-enable
stc-status

Description

These commands are for advanced usage only. They allow the user to control the usage of Simics-internal caches.

The Simulator Translation Caches (STCs) are designed to increase execution performance. The D-STC caches data translations (logical to physical to real (host) address). The I-STC caches instruction translations of taken jumps. Finally, the IO-STC caches logical to physical address translation for device accesses.

By default the STCs are **on**.

When a memory hierarchy is connected (such as a cache module) it must have been designed to work along with the STCs, or it may not be called for all the memory transactions it is interested in. These commands can be used to detect if too many translations are kept in the STCs, causing the simulation to be incorrect.

Turning the STCs off means that current contents will be flushed and no more entries will be inserted into the STCs.

Provided By

[Simics Core](#)

iostc-enable**Synopsis**

iostc-enable
dstc-disable
dstc-enable
iostc-disable
istc-disable
istc-enable
stc-status

Description

These commands are for advanced usage only. They allow the user to control the usage of Simics-internal caches.

The Simulator Translation Caches (STCs) are designed to increase execution performance. The D-STC caches data translations (logical to physical to real (host) address). The I-STC caches instruction translations of taken jumps. Finally, the IO-STC caches logical to physical address translation for device accesses.

By default the STCs are **on**.

When a memory hierarchy is connected (such as a cache module) it must have been designed to work along with the STCs, or it may not be called for all the memory transactions it is interested in. These commands can be used to detect if too many translations are kept in the STCs, causing the simulation to be incorrect.

Turning the STCs off means that current contents will be flushed and no more entries will be inserted into the STCs.

Provided By
[Simics Core](#)

istc-disable

Synopsis

```
istc-disable
dstc-disable
dstc-enable
iostc-disable
iostc-enable
istc-enable
stc-status
```

Description

These commands are for advanced usage only. They allow the user to control the usage of Simics-internal caches.

The Simulator Translation Caches (STCs) are designed to increase execution performance. The D-STC caches data translations (logical to physical to real (host) address). The I-STC caches instruction translations of taken jumps. Finally, the IO-STC caches logical to physical address translation for device accesses.

By default the STCs are **on**.

When a memory hierarchy is connected (such as a cache module) it must have been designed to work along with the STCs, or it may not be called for all the memory transactions it is interested in. These commands can be used to detect if too many translations are kept in the STCs, causing the simulation to be incorrect.

Turning the STCs off means that current contents will be flushed and no more entries will be inserted into the STCs.

Provided By
[Simics Core](#)

istc-enable

Synopsis

```
istc-enable
```

dstc-disable
dstc-enable
iostc-disable
iostc-enable
istc-disable
stc-status

Description

These commands are for advanced usage only. They allow the user to control the usage of Simics-internal caches.

The Simulator Translation Caches (STCs) are designed to increase execution performance. The D-STC caches data translations (logical to physical to real (host) address). The I-STC caches instruction translations of taken jumps. Finally, the IO-STC caches logical to physical address translation for device accesses.

By default the STCs are **on**.

When a memory hierarchy is connected (such as a cache module) it must have been designed to work along with the STCs, or it may not be called for all the memory transactions it is interested in. These commands can be used to detect if too many translations are kept in the STCs, causing the simulation to be incorrect.

Turning the STCs off means that current contents will be flushed and no more entries will be inserted into the STCs.

Provided By

[Simics Core](#)

license**Synopsis**

license [-third-party]

Description

Prints information about the license that applies to this copy of Simics. The **-third-party** flag enables output of licenses and copyrights for all third party code that is included with the Simics distribution.

Provided By

[Simics Core](#)

list**Synopsis**

list [-s|-d] [address|“location”] [maxlines]

Description

List the source code corresponding to a given address, function or line. The location

can be specified as *line* or *file : line* (list from that line); *function* or *file : function* (list that function); or *address* (list from that address).

At most *maxlines* lines of source or asm are printed. *-s* produces source intermixed with disassembly, and *-d* disassembly only.

Provided By

[Simics Core](#)

See Also

[disassemble](#), [whereis](#), [pos](#), [symval](#), [sym-source](#), [load-symbols](#), [sym-value](#)

list-attributes

Synopsis

list-attributes “*class*” [“*attribute-name*”] [-i]

Description

Print a list of all attributes that are registered for a class. For every attribute the type, as well as additional flags are listed. See the *SIM_register_typed_attribute()* documentation function for valid attribute types and flags. If an attribute name is given, the description for that particular attribute will be displayed. Attributes marked as internal will only be listed when the *-i* flag is used. For backward compatibility the name of an existing object can be used instead of a class name.

Provided By

[Simics Core](#)

list-bookmarks

Synopsis

list-bookmarks

Description

Lists time bookmarks.

Provided By

[Simics Core](#)

See Also

[set-bookmark](#), [delete-bookmark](#)

list-breakpoints

Alias

ib, info-breakpoints

Synopsis

list-breakpoints [-all]
get-breakpoint-list [-all]

Description

Prints information about all breakpoints set. The following information is printed for memory breakpoints: the id (used by other commands to refer to the breakpoint), if the breakpoint is set on physical or virtual addresses and the access type (r = read, w = write, or x = execute), if enabled (see the **enable** command), the address range of the breakpoint, how many times the breakpoint has been triggered, and what memory space or context object it is set in.

If prefix, substring and/or pattern conditions are set it will be printed as well (see **set-prefix**, **set-substr** and **set-pattern** command).

Time breakpoints are also listed.

If **-all** is passed as argument, **list-breakpoints** will also list all internal breakpoints set for simulation purposes.

The **get-breakpoint-list** return a list of all breakpoint numbers instead of printing information about the breakpoints.

Provided By

[Simics Core](#)

See Also

[`<breakpoint>.break, delete, enable, ignore, set-prefix, set-substr, set-pattern`](#)

list-checkpoints**Synopsis**

list-checkpoints [path] [-r]

Description

List all checkpoints in the *path* directory or in the current directory if no path argument given. Use *-r* to search for checkpoints recursively in all sub-directories to *path*.

Provided By

[Simics Core](#)

See Also

[`read-configuration`](#)

list-classes**Synopsis**

list-classes [-l]
get-class-list [-l]

Description

Print (**list-classes**) or return (**get-class-list**) a list of all available configuration classes. The **-l** flag will reduce the list to classes that has been registered by loaded modules.

Provided By

[Simics Core](#)

list-components**Synopsis**

```
list-components [component] ["class"] [-s] [-v] [-t] [-all] [-recursive]
get-component-list [component] ["class"] [-t] [-all] [-recursive]
```

Description

Lists components with their names, types, and connectors; and for each connector, the destination component and connector. With the *class* argument, you can restrict the listing to components of a particular class.

By default, only the components in the current namespace are listed (see the **change-namespace** command). If no current namespace has been selected, only top-level components are shown. The *component* argument can be used to override the current namespace.

With the **-all** flag, list all components, regardless of where they live. With the **-recursive** flag, in addition to listing all components that are in the selected namespace, also list all components in the namespaces below it.

Note: If the *legacy_conf* preferences option is set to true, all components belong to the top level, so most of these options have no effect.

The **-s** and **-v** flags select a briefer listing (the default), and a more verbose listing, respectively. With **-t**, list only top-level components.

The **get-component-list** command is similar to **list-components**, but returns a list of components instead of printing them. It does not recognize the **-s** and **-v** flags.

Provided By

[Simics Core](#)

list-debug-contexts**Synopsis**

```
list-debug-contexts ["context-query"]
```

Description

List debug contexts matching the context query *context-query*. The displayed value is the list of all the fully qualified names of the context matching the context query. The return value is a list of contexts. Each element of the list is a list of two elements: the context id and the fully qualified context name.

context-query defaults to * which matches all contexts.

Provided By
[tcf-agent](#)

list-directories

Synopsis
list-directories

Description

Print a list of all directories in the Simics search path.

Provided By
[Simics Core](#)

See Also
[add-directory](#)

list-failed-modules

Alias
 module-list-failed

Synopsis
list-failed-modules [“*substr*”] [-v]

Description

Lists the modules (Simics extensions) that are not loadable, optionally only those matching *substr*.

Similar to **list-modules** but shows modules that will not load into Simics, and the reason why Simics refuses to load them (e.g., missing symbol, wrong version, ...).

If the **-v** flag is specified, show verbose information, with the full path to the module file and any library loader error message.

The **MODULE** column contains the name of the module or the filename of the shared library file if the module name could not be established.

If the module has the same name as another module, an **X** will be printed in the **DUP** column.

If the module could not be loaded since it was compiled or written for a different version of Simics, the version it was built for will be printed in the **VERSION** column.

The **USR_VERS** will contain the user version string, if provided.

The **LINK** column contains any linker error (cf. the **dllerror(3)** manpage).

When the **-v** flag is provided, the **PATH** column will contain linker information for the module.

Provided By
[Simics Core](#)

See Also

[list-modules](#), [module-list-refresh](#), [load-module](#), [add-module-directory](#)

list-hap-callbacks**Synopsis**

list-hap-callbacks [“*hap*”]

Description

This command is deprecated.

Prints a list of all callbacks installed for *hap*, or for all haps if the argument is omitted.

Provided By

[Simics Core](#)

list-haps**Alias**

hl

Synopsis

list-haps [“*hap*”]

Description

Prints a description of *hap*. If the name is omitted a list of all haps will be printed.

Provided By

[Simics Core](#)

list-hypersim-patterns**Synopsis**

list-hypersim-patterns [-u]

Description

List which hypersim-pattern classes that are available. With the *-u* flag it also shows which hypersim-pattern-matcher that currently uses the pattern.

Provided By

[hypersim-pattern-matcher](#)

list-length**Synopsis**

list-length *list*

Description

Returns the length of a CLI list.

Provided By

[Simics Core](#)

list-modules**Alias**

module-list

Synopsis

list-modules [“*substr*”] [-l] [-v]

Description

Lists all modules that can be loaded into Simics. If the optional *substr* argument is specified, only modules with a matching name will be printed.

Use -v to get more information on the modules, and -l to only list loaded modules.

The “ABI” column is the oldest build ID of Simics that the module is supposed to work with. “Build” is the module-specific build ID, defaulting to the build-id of the Simics package that the module was compiled with.

The “Thread-safe” column shows whether the module has been certified as thread safe. Modules written in Python do not need to be certified and are marked with a hyphen (-).

Provided By

[Simics Core](#)

See Also

[list-failed-modules](#), [module-list-refresh](#), [load-module](#), [add-module-directory](#)

list-namespaces**Synopsis**

list-namespaces [-n]

Description

Lists all namespaces (objects) and which classes or interfaces they belong to. A namespace is the same as a configuration object. Many objects implement commands local to them. These commands are invoked by giving the object name followed by a period and then the local command name; e.g., `rec0.playback-start`.

If the -n flag is given, the output will be sorted on the object name instead of the class name, which is the default.

Some objects also implement command interfaces. A command interface is a collection of commands that can be used by an object implementing this interface. For example,

breakpoint is an interface that is implemented by objects of the **memory-space** class. This allows one to write **phys_mem0.break 0xffc00** to set a breakpoint in the memory interface.

Objects implementing command interfaces are listed in the second half of output from this command.

Provided By
[Simics Core](#)

See Also
[list-objects](#)

list-objects

Synopsis

```
list-objects ["type"] [component] [-n] [-a] [-all] [-recursive]
get-object-list ["type"] [component] [-all] [-recursive]
```

Description

Lists configuration objects and the class they belong to. With the *type* argument, you can restrict the listing to objects of a particular class, or to objects that implement a particular interface.

By default, only the objects in the current namespace are listed (see the **change-namespace** command). The *component* argument can be used to override the current namespace.

Use the **-recursive** flag to list objects in both the current namespace and in the namespaces below it in the hierarchy.

With the **-all** flag, all objects are listed, regardless of namespace.

The objects are sorted by class name by default. The **-n** flag sorts them by name instead.

Component objects are printed first and then all other objects. To mix all objects in the same list, use the **-a** flag.

The **get-object-list** command is similar to **list-objects**, but returns a list of object names instead of printing them. It does not recognize the **-n** and **-a** flags.

Provided By
[Simics Core](#)

See Also
[list-namespaces](#), [change-namespace](#)

list-objects-with-interface

Synopsis

```
list-objects-with-interface "interface" [component] [-all] [-recursive]
```

Description

Lists configuration objects implementing specific interface.

By default, only the objects in the current namespace are listed (see the **change-namespace** command). The *component* argument can be used to override the current namespace.

Use the **-recursive** flag to list objects in both the current namespace and in the namespaces below it in the hierarchy.

With the **-all** flag, all objects are listed, regardless of namespace.

Provided By

[Simics Core](#)

list-port-forwarding-setup**Synopsis**

list-port-forwarding-setup

Description

Lists the current port forwarding and NAPT configuration.

Provided By

[service-node](#)

See Also

[connect-real-network](#), [connect-real-network-napt](#), [connect-real-network-port-in](#), [connect-real-network-port-out](#)

list-preferences**Synopsis**

list-preferences [-v]

Description

Print the current preferences. Use the **-v** flag to get verbose output that includes a description of each preference entry.

Provided By

[Simics Core](#)

See Also

[save-preferences](#)

list-script-branches**Synopsis**

list-script-branches

Description

List all currently active script branches.

Provided By

[Simics Core](#)

See Also

[script-branch](#), [interrupt-script-branch](#)

list-sections**Synopsis**

list-sections *symbol-file*

Description

Lists the sections of the symbol file *symbolfile*.

symbolfile uses Simics's Search Path and path markers (%simics%, %script%) to find the symbol file. Refer to *The Command Line Interface* chapter of the *Hindsight User's Guide* manual for more information on how Simics's Search Path is used to locate files.

Provided By

[tcf-agent](#)

list-segments**Synopsis**

list-segments *symbol-file*

Description

Lists the segments of the symbol file *symbol-file*.

symbol-file uses Simics's Search Path and path markers (%simics%, %script%) to find the symbol file. Refer to *The Command Line Interface* chapter of the *Hindsight User's Guide* manual for more information on how Simics's Search Path is used to locate files.

Provided By

[tcf-agent](#)

list-variables**Alias**

`list-vars`

Synopsis

list-variables

Description

Lists all CLI variables and their current values. CLI variables can be used to store temporary values. To set a variable, write `$variable = value` at the Simics prompt. The value can be of type integer, string, float, boolean or a list or values. To access a variable, prefix the name with a dollar sign (\$); e.g., `$variable`. A variable can be used wherever an expression can be used. For example:

```
simics> $tmp = %pc + 4
simics> $count = 10
simics> disassemble $tmp $count
```

They can also be accessed from Python by using the name space `simenv`:

```
simics> $foo = 1 + 4 * 4
simics> @print simenv.foo
17
simics> @simenv.bar = "hello"
simics> echo $bar
hello
```

Provided By

[Simics Core](#)

See Also

[\\$, read-variable, defined](#)

load-binary**Synopsis**

```
load-binary filename [offset] [-v] [-pa] [-l] [-n]
<memory-space>.load-binary filename [offset] [-v] [-pa] [-n]
```

Description

Load a binary (executable) file into the given physical or virtual memory space. The supported formats are ELF, Motorola S-Record, PE32 and PE32+.

By default the virtual load address from the file is used. The physical load address can be used instead, for file formats supporting both, by specifying the `-pa` flag. The load address selected does not affect if the binary is loaded into the virtual or physical address space.

If an *offset* is supplied, it will be added to the load address taken from the file.

The global **load-binary** command will use the currently selected processor to find the memory space to load the binary into. If the `-l` flag is given, it will load it into the virtual memory space, otherwise it will use the physical memory space. The processor must have a valid virtual to physical translation set up.

When using the namespace command on a **processor** object, it will load the binary into the virtual memory space of that processor.

When using the namespace command on a **memory-space** object, it will load the binary directly into that memory space without any virtual to physical translation.

The **-v** flag turns on verbose mode, printing information about the loaded file.

The **-n** flags tells the command to not clear .bss areas in the file.

The return value is the address of the execution entry point. This value is typically used in a call to **set-pc**.

load-binary uses Simics's Search Path and path markers (%simics%, %script%) to find the file to load. Refer to *The Command Line Interface* chapter of the *Hindsight User's Guide* manual for more information on how Simics's Search Path is used to locate files.

Provided By

[Simics Core](#)

See Also

[load-file](#), [add-directory](#)

load-file

Synopsis

```
load-file filename [offset]
<memory-space>.load-file filename [offset]
```

Description

Loads the contents of the file named *filename* into memory (defaulting to the current frontend processor's physical memory space), starting at physical address *offset*. Default offset is 0.

load-file uses Simics's Search Path and path markers (%simics%, %script%) to find the file to load. Refer to *The Command Line Interface* chapter of the *Hindsight User's Guide* manual for more information on how Simics's Search Path is used to locate files.

Provided By

[Simics Core](#)

See Also

[load-binary](#), [add-directory](#)

load-module

Synopsis

```
load-module "module"
```

Description

Load a module (Simics extension). Simics supports dynamically loadable modules. Read the *Simics Model Builder User's Guide* for more info on how to write modules.

Provided By[Simics Core](#)**See Also**[list-modules](#), [list-failed-modules](#), [module-list-refresh](#), [add-module-directory](#)**load-persistent-state****Synopsis****load-persistent-state** *file* [“*prefix*”]**Description**

Load persistent simulator state from *file*. Persistent data typically includes disk images, NVRAM and flash memory contents and clock settings, i.e. data that survive reboots. The *prefix* argument can be used to add a name prefix to all objects in the persistent state file.

Provided By[Simics Core](#)**See Also**[save-persistent-state](#), [read-configuration](#)**local****Synopsis****local** *\$foo* = *value***Description**

Makes a CLI variable local in an assignment.

Local variables only exist until the end of the current command block.

Provided By[Simics Core](#)**See Also**[=](#), [\\$](#)**log****Synopsis****log** [*count*]**Description**

Display entries in the log buffers. The namespace version displays the entries for a specific object, while the global **log** command lists the entries of all object’s log buffers but sorted by time. The optional argument is the number of entries to list. Only the last 10 entries are listed by default.

All log buffers are zero-sized by default. The **log-size** command has to be used to allocate space for log entries. This has to be done before running the part of the simulation that generates the log messages.

Only log messages that match the currently configured level, group and type are saved to the log buffers.

Provided By
[Simics Core](#)

See Also

[`<conf_object>.log-group`](#), [`log-level`](#), [`log-size`](#), [`log-type`](#)

log-level

Synopsis

log-level [*level*] [“*class*”] [-all]

Description

This command is used to get or set the log level of all objects in the current namespace, or globally if `-all` is set.

If the *class* argument is specified, only operate on objects of that class.

Objects in Simics can generate log messages on different *log levels*. These messages will be shown in the Simics command line window if the log level for the object has been set high enough.

The default level is 1, and this is the lowest level that objects can report messages on. Setting it to 0 will inhibit output of all messages except error messages.

Messages are also added to an access log that can be viewed by the **log** command in Simics.

There are four log levels defined:

- 1 - important messages printed by default
- 2 - “high-level” informative messages
- 3 - standard debug messages
- 4 - detailed information, such as register accesses

The namespace version of this command, [`<conf_object>.log-level`](#), sets the log level on all objects of the specified component. The `-r` flag recursively updates all sub-components and their objects.

Provided By
[Simics Core](#)

See Also

[`log`](#), [`<conf_object>.log-group`](#), [`log-size`](#), [`log-type`](#), [`change-namespace`](#)

log-setup

Synopsis

```
log-setup [-time-stamp] [-no-time-stamp] [-console] [-no-console] [-no-log-file] [-overwrite]
[logfile]
```

Description

This command controls the log output generated by log objects and trace commands.

The `-time-stamp` flag will cause future log output to include a time stamp: the name of the current processor, its current program counter value, and its current cycle count. If the current “processor” does not execute instructions (such as the `clock`), the program counter value is omitted. Time stamps are disabled with `-no-time-stamp`.

A file that receives all log output can be specified with the `logfile` argument. `-no-log-file` disables an existing log file. To overwrite an existing file, the `-overwrite` flag has to be given.

The `-console` and `-no-console` arguments turn log output to the command line console on or off.

Any messages printed at a level below or equal to the current log level (see the **log-level** command) will always be stored in the in-memory log buffer (see the **log** command).

Provided By

[Simics Core](#)

See Also

[log](#), [<conf_object>.log-group](#), [log-size](#), [log-type](#), [<conf_object>.wait-for-log](#)

log-size**Synopsis**

```
log-size [size]
```

Description

The namespace version of this command changes the buffer size (number of entries) for log messages and I/O trace entries for an objects. The global command applies to all log objects. When called with no arguments, the size of the log buffers are listed.

Provided By

[Simics Core](#)

See Also

[log](#), [<conf_object>.log-group](#), [log-level](#), [log-type](#)

log-type**Synopsis**

```
log-type [-add] [-sub] [“log-type”]
```

Description

Log messages are categorised into one of the several log types. By default, messages of all types are handled in the same way. This command can be used to select one or several types. Only messages of the selected types will be logged and displayed, as defined by the **log-level** command. The flags **-add** and **-sub** can be used to add and remove a single log type. The log types are documented with the **log_type_t** data type, and are Info, Error, Undefined, Spec_Violation, Unimplemented. All types can be enabled by setting **log-type** to **all**.

Provided By

[Simics Core](#)

See Also

[log](#), [<conf_object>.log-group](#), [log-level](#), [log-size](#)

logical-to-physical**Alias**

l2p

Synopsis

logical-to-physical [*cpu-name*] *address*

Description

Translate the given logical *address* to a physical one. The operation is performed as data read from processor *cpu-name*.

For x86 CPUs, a logical address can be specified as **<segment register>:<offset>** or **l:<linear address>**. If no prefix is given **ds:<offset>** will be assumed.

If the CPU is omitted the current CPU will be used.

No side-effects will be triggered; e.g., if the translation is not in the TLB.

Provided By

[Simics Core](#)

ls**Synopsis**

ls [*path*]

Description

List files in working directory of Simics. Works like the **ls** command in a Unix shell but with a single optional parameter, the directory to list files in.

Provided By

[Simics Core](#)

See Also[cd](#), [pwd](#)**magic-breakpoint-enabled****Synopsis**

magic-breakpoint-enabled
disable-magic-breakpoint
enable-magic-breakpoint

Description

The **enable-magic-breakpoint** command sets a breakpoint on the magic instruction with argument 0, except on SPARC targets. On SPARC targets it is a magic instruction with the top 6 bits of the 22-bit parameter field set to binary 000100. The breakpoint can be removed with the **disable-magic-breakpoint** command. To break on all magic instructions, use `break-hap Core_Magic_Instruction`.

Provided By[Simics Core](#)**match-string****Synopsis**

match-string "*pattern*" "*string*" [-error]

Description

Match *pattern* with *string* returning TRUE if the pattern matches and FALSE if not. The command will signal an error if the *-error* flag is used and the string does not match the pattern. The format of *pattern* is the same as used by the Python `re.search()` method. If the pattern contains groups, the return value on a match is a list of matching strings. Example:

```
match-string "abd" "abcdef" → FALSE
match-string "abc" "abcdef" → TRUE
match-string "(a*)(([0-9]*)x" "aaa04x" → ["aaa", "04"]
```

Provided By[Simics Core](#)**See Also**[split-string](#)**max****Synopsis**

max arg1 arg2

Description

Returns the larger value of *arg1* and *arg2*.

Provided By

[Simics Core](#)

min**Synopsis**

min *arg1* *arg2*

Description

Returns the smaller value of *arg1* and *arg2*.

Provided By

[Simics Core](#)

module-list-refresh**Synopsis**

module-list-refresh

Description

Refresh (reload) the list of all Simics modules.

This command causes Simics to re-query all modules currently not loaded. This can be used after changing or adding a module that Simics, when started, considered as non-loadable.

Provided By

[Simics Core](#)

See Also

[list-modules](#), [list-failed-modules](#), [load-module](#)

move-object**Synopsis**

move-object *src* “*dst*”

Description

Move object location from *src* to *dst*.

Provided By

[Simics Core](#)

native-path

Synopsis

native-path “*filename*”

Description

Converts a path to its host native form. On Unix, this command returns *filename* unchanged. On Windows, it translates Cygwin-style paths to native Windows paths. Refer to the documentation *SIM_native_path()*, for a detailed description of the conversions made.

This command can be used for portability when opening files residing on the host filesystem.

Provided By

[Simics Core](#)

network-helper**Synopsis**

network-helper [*helper*]

Description

Sets the file name of the helper executable that opens the host network interface for real-network connections using raw access. This helper, **openif** in the distribution, needs privileges to run and should be installed in such a way that it is run as root, typically by installing it as setuid root. Without arguments, displays the current setting.

This command has no effect under Windows.

Provided By

[real-network](#)

new-attr-meter**Synopsis**

new-attr-meter [“*name*”] “*attribute*” [*realtime-period*] [*simtime-period*]

Description

Create a new attribute meter, which polls the given attribute (specified as *object-name.attributename*) periodically, making it possible to visualize it in the GUI’s Statistics Plot window. The attribute must be numerical (i.e., have type **i** or **f**).

realtime-period and *simtime-period* determines how often to poll the attribute, in seconds of real and simulated time, respectively.

Provided By

[attr-meter](#)

new-cell-and-clocks

Synopsis

```
new-cell-and-clocks ["name"] [clock_number] [freq_mhz] [domain]
```

Description

Creates an instantiated component of the class “cell-and-clocks”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[clock](#)

See Also

[create-cell-and-clocks](#)

new-central-server**Synopsis**

```
new-central-server [port] [file] [min-latency]
```

Description

Create a Simics Central server object. The server will by default listen to Simics Central client connections on TCP port 1909 and, on systems supporting it, UNIX file socket /tmp/simics-central.*user*. It will also accept connections from a Central client in the same Simics process.

The *port* argument can be used to change the TCP listen port. By setting it to -1, the server will not listen for TCP connections. By setting it to 0, the server will choose an available TCP port number. The used port number will be printed when the server successfully opened the port.

The *file* argument can be used to change the file name for the UNIX file socket. An empty string disables it.

The *min-latency* argument specifies the minimum latency in nanoseconds for inter-simics communication enforced by Simics Central.

Provided By

[central](#)

new-context**Synopsis**

```
new-context ["name"]
```

Description

Create a new context object called *name*. The context is initially not bound to any processor.

Provided By

[Simics Core](#)

See Also

[set-context](#), <context>.symtable

new-cp3-quad100tx**Synopsis**

new-cp3-quad100tx [“*name*”] “*mac_addr1*” “*mac_addr2*” “*mac_addr3*” “*mac_addr4*”

Description

This command creates an instantiated component of the class **cp3_quad100tx**.

The class description for the **cp3_quad100tx** class: The cp3-quad100tx component class.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***mac_addr1* is Required**

The MAC address of eth1

***mac_addr2* is Required**

The MAC address of eth2

***mac_addr3* is Required**

The MAC address of eth3

***mac_addr4* is Required**

The MAC address of eth4

Provided By

[cp3_quad100tx](#)

new-datagram-link**Synopsis**

new-datagram-link [“*name*”] [“*global_id*”] [*goal_latency*]

Description

This command creates an instantiated component of the class **datagram_link**.

The class description for the **datagram_link** class: The datagram link component creates a datagram-link, which is a simple broadcast bus forwarding messages (as sequences of bytes) from a sender device to all other devices present of the link. The datagram-link is both an example of how to build a link with the Simics Link Library, and a simple broadcast link that can be reused when multi-cell communication between devices is necessary. Refer to the *Link Library Programming Guide* for more information.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***global_id* is Optional**

Global identifier for use in distributed simulation or NIL if the link is not distributed.

***goal_latency* is Optional**

Goal latency in seconds for this link. See also the **set-min-latency** command.

Provided By

[datagram-link](#)

new-dummy-component**Synopsis**

new-dummy-component [“*name*”] [*domain*]

Description

Creates an instantiated component of the class “dummy-component”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

See Also

[create-dummy-component](#)

new-etg — deprecated**Synopsis**

new-etg [“*name*”] *link* “*ip*” “*netmask*” “*target*” [“*gw*”] [*idx*] [*pps*] [*size*] [*port*]

Description

This command is deprecated; use **new-std-etg** instead.

Create a new **etg** object.

Provided By

[etg](#)

new-etg-comp**Synopsis**

new-etg-comp [“*name*”] “*dst_ip*” [“*gateway_ip*”] “*ip*” [“*mac_address*”] “*netmask*” [*packet_size*] [*port*] [*pps*]

Description

This command creates an instantiated component of the class **etg_comp**.

The class description for the **etg_comp** class: The etg component represents an Ethernet traffic generator.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***dst_ip* is Required**

Destination IP address for generated traffic.

***gateway_ip* is Optional**

'Gateway for non-local traffic.

***ip* is Required**

IP address of the traffic generator.

***mac_address* is Optional**

The MAC address of the traffic generator.

***netmask* is Required**

IP netmask of the traffic generator.

***packet_size* is Optional**

Packet size.

***port* is Optional**

Port.

***pps* is Optional**

Traffic rate in packets per second.

Provided By

[std-components](#)

new-etg-panel**Synopsis**

new-etg-panel [“*name*”]

Description

This command creates an instantiated component of the class **etg_panel**.

The class description for the **etg_panel** class: The Ethernet Generator System Panel.

***name* is Optional**

If not specified, the component will get a class-specific default name.

Provided By

[std-components](#)

new-eth-injector-comp**Synopsis**

new-eth-injector-comp [“*name*”]

Description

This command creates an instantiated component of the class **eth_injector_comp**.

The class description for the **eth_injector_comp** class: The Ethernet frame injector is a pseudo-device that reads a pcap formatted file and inject the packets it found into another device, or an Ethernet link.

***name* is Optional**

If not specified, the component will get a class-specific default name.

Provided By

[eth_injector_comp](#)

new-ethernet-cable**Synopsis**

new-ethernet-cable [“*name*”] [“*global_id*”] [*goal_latency*]

Description

This command creates an instantiated component of the class **ethernet_cable**.

The class description for the **ethernet_cable** class: Ethernet cable: this component represents a two-points ethernet cable, allowing two devices to connect to each other.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***global_id* is Optional**

Global identifier for use in distributed simulation or NIL if the link is not distributed.

***goal_latency* is Optional**

Goal latency in seconds for this link. See also the **set-min-latency** command.

Provided By

[eth-links](#)

new-ethernet-hub**Synopsis**

new-ethernet-hub [“*name*”] [“*global_id*”] [*goal_latency*]

Description

This command creates an instantiated component of the class **ethernet_hub**.

The class description for the **ethernet_hub** class: Ethernet hub: this component represents a simple broadcasting ethernet link allowing any number of devices to connect.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***global_id* is Optional**

Global identifier for use in distributed simulation or NIL if the link is not distributed.

***goal_latency* is Optional**

Goal latency in seconds for this link. See also the **set-min-latency** command.

Provided By
[eth-links](#)

new-ethernet-switch

Synopsis

new-ethernet-switch [“*name*”] [“*global_id*”] [*goal_latency*]

Description

This command creates an instantiated component of the class **ethernet_switch**.

The class description for the **ethernet_switch** class: Ethernet switch: this component represents a switched ethernet network, allowing any number of devices to connect and optimizing the packet routing according to what is learned about the MAC addresses talking on the link.

name is Optional

If not specified, the component will get a class-specific default name.

global_id is Optional

Global identifier for use in distributed simulation or NIL if the link is not distributed.

goal_latency is Optional

Goal latency in seconds for this link. See also the **set-min-latency** command.

Provided By
[eth-links](#)

new-ethernet-vlan-switch

Synopsis

new-ethernet-vlan-switch [“*name*”] [“*global_id*”] [*goal_latency*]

Description

This command creates an instantiated component of the class **ethernet_vlan_switch**.

The class description for the **ethernet_vlan_switch** class: Ethernet VLAN switch: this component represents a switched ethernet network with VLAN support. Any number of devices is allowed to connect to various ports of the switch. Each port can be configured with its own VLAN information, in order to create sub-networks in the switch.

name is Optional

If not specified, the component will get a class-specific default name.

global_id is Optional

Global identifier for use in distributed simulation or NIL if the link is not distributed.

***goal_latency* is Optional**

Goal latency in seconds for this link. See also the **set-min-latency** command.

Provided By

[eth-links](#)

new-example-keypad**Synopsis**

new-example-keypad [“*name*”]

Description

Creates an instantiated component of the class “example-keypad”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

See Also

[create-example-keypad](#)

new-example-status-panel**Synopsis**

new-example-status-panel [“*name*”]

Description

Creates an instantiated component of the class “example-status-panel”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

See Also

[create-example-status-panel](#)

new-file-cdrom**Synopsis**

new-file-cdrom *file* [“*name*”]

Description

Create a new file-cdrom object from *file* (which should be a valid CD-ROM (ISO) image), named *name*. If *name* is not given, an object name is derived from the file name. This object can then be inserted into a simulated CD-ROM device using the <**device**>.insert command.

Provided By[file-cdrom](#)**new-gdb-remote****Synopsis****new-gdb-remote** [“*name*”] [*port*] [*cpu*] [“*architecture*”] [*context*]**Description**

Starts listening to incoming connection requests from GDB sessions (provided that a configuration has been loaded). Simics will listen to TCP/IP requests on the port specified by *port*, or 9123 by default. If *port* is set to zero, an arbitrary free port will be selected.

You can either attach the GDB session to a particular processor, or to a particular context object. If you specify a processor (with the *cpu* argument), the GDB session will follow the execution on that particular processor. It will see all code that runs on that processor: user processes, operating system, hypervisor, everything.

If instead you specify a context (with the *context* argument), the GDB session will follow that context; this is useful in combination with process tracking, which can make a context follow a specific process as it moves between processors, is descheduled, etc. The end result is that GDB sees that process no matter which process it runs on, and does not see other processes or the operating system.

The *architecture* argument can be used to specify a particular architecture for the GDB session. It should be the architecture name used by Simics and not the GDB architecture name. For example, if you are debugging a 32-bit program on a 64-bit x86 processor, you may want to specify `x86` as *architecture* and run `set architecture i386` in GDB before connecting. For 64-bit PowerPC platforms set this argument to `ppc32` to debug a 32-bit program. If not given, the architecture of the specified processor will be used, or the architecture of the processor attached to the specified context.

In GDB, use the command `target remote host:port` to connect to Simics.

Provided By[gdb-remote](#)**new-generic-pcie-switch****Synopsis****new-generic-pcie-switch** [“*name*”] [*device_id*] [*port_count*] [*vendor_id*]**Description**

This command creates an instantiated component of the class **generic_pcie_switch**.

The class description for the **generic_pcie_switch** class: Generic PCIe switch with a configurable number of ports and configurable vendor and device IDs

***name* is Optional**

If not specified, the component will get a class-specific default name.

***device_id* is Optional**

Device ID

***port_count* is Optional**

Number of down ports

***vendor_id* is Optional**

Vendor ID

Provided By

[generic-pcie-switch-comp](#)

new-glink**Synopsis**

new-glink [*"name"*] [*latency*]

Description

Create a new generic message link. A default name and latency are provided if they are not specified on the command line.

Provided By

[g-link](#)

new-hap-meter**Synopsis**

new-hap-meter [*"name"*]

Description

Create a new hap meter. The new hap meter can then be instructed to collect statistics on the occurrence of various haps, and the result visualized in the GUI's Statistics Plot window.

Provided By

[hap-meter](#)

See Also

[`<hap-meter>.listen-for-exceptions`](#), [`<hap-meter>.listen-for-hap`](#)

new-i2c-link-v2**Synopsis**

new-i2c-link-v2 [*"name"*] [*"global_id"*] [*goal_latency*]

Description

This command creates an instantiated component of the class **i2c_link_v2**.

The class description for the **i2c_link_v2** class: This component represents a simple i2c link allowing any number of devices to connect.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***global_id* is Optional**

Global identifier for use in distributed simulation or NIL if the link is not distributed.

***goal_latency* is Optional**

Goal latency in seconds for this link. See also the **set-min-latency** command.

Provided By

[i2c-link-v2](#)

new-mem-traffic-meter**Synopsis**

new-mem-traffic-meter [“*name*”]

Description**Provided By**

[mem-traffic-meter](#)

new-micron-mtfc2ggqdi-emmc-card**Synopsis**

new-micron-mtfc2ggqdi-emmc-card [“*name*”] [“*file*”]

Description

This command creates an instantiated component of the class **micron_mtfc2ggqdi_emmc_card**.

The class description for the **micron_mtfc2ggqdi_emmc_card** class: A 2GB Micron eMMC card.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***file* is Optional**

File with disk contents for the full disk. Either a raw file or a CRAFF file

Provided By

[mmc-card-components](#)

new-micron-mtfc4ggqdi-emmc-card**Synopsis**

```
new-micron-mtfc4ggqdi-emmc-card ["name"] ["file"]
```

Description

This command creates an instantiated component of the class **micron_mtfc4ggqdi_emmc_card**.

The class description for the **micron_mtfc4ggqdi_emmc_card** class: A Micron MTFC4GGQDI-IT eMMC Card.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***file* is Optional**

File with disk contents for the full disk. Either a raw file or a CRAFF file

Provided By

[mmc-card-components](#)

new-micron-mtfc4ggqdi-sdhc-card**Synopsis**

```
new-micron-mtfc4ggqdi-sdhc-card ["name"] ["file"]
```

Description

This command creates an instantiated component of the class **micron_mtfc4ggqdi_sdhc_card**.

The class description for the **micron_mtfc4ggqdi_sdhc_card** class: A SDHC card component similar to Micron MTFC4GGQDI-IT eMMC.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***file* is Optional**

File with disk contents for the full disk. Either a raw file or a CRAFF file

Provided By

[mmc-card-components](#)

new-os-awareness**Synopsis**

```
new-os-awareness ["name"]
```

Description

This command creates an instantiated component of the class **os_awareness**.

The class description for the **os_awareness** class: The “os-awareness” component tracks software running on the system.

***name* is Optional**

If not specified, the component will get a class-specific default name.

Provided By

[os-awareness](#)

new-phy-comp**Synopsis**

new-phy-comp [“*name*”] [*mii_address*] [*phy_id*]

Description

This command creates an instantiated component of the class **phy_comp**.

The class description for the **phy_comp** class: Component representing a generic IEEE 802.3 PHY

***name* is Optional**

If not specified, the component will get a class-specific default name.

***mii_address* is Optional**

PHY address on MII bus

***phy_id* is Optional**

PHY ID (i.e., vendor)

Provided By

[phy-comp](#)

new-rapidio-link**Synopsis**

new-rapidio-link [“*name*”] [“*global_id*”] [*goal_latency*]

Description

This command creates an instantiated component of the class **rapidio_link**.

The class description for the **rapidio_link** class: RapidIO link connecting two RapidIO devices

***name* is Optional**

If not specified, the component will get a class-specific default name.

***global_id* is Optional**

Global identifier for use in distributed simulation or NIL if the link is not distributed.

***goal_latency* is Optional**

Goal latency in seconds for this link. See also the **set-min-latency** command.

Provided By
[rapidio-link](#)

new-rapidio-tape

Synopsis

new-rapidio-tape [*playback-file*] [*record-file*] [“*prefix*”] [*peer*]

Description

Starts generating rapidio traffic from a specified file. The file should consist of cycle and data information, try record some traffic to see an example.

Provided By
[rapidio-tape](#)

See Also

[`<rapidio_tape>.playback-stop`](#)

new-real-network-bridge

Synopsis

new-real-network-bridge [“*name*”] [“*interface*”] [*persistent*] [*tap_bridge*] “*access*” *no_mac_translate*

Description

Creates an instantiated component of the class “real-network-bridge”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[real-network](#)

See Also

[`create-real-network-bridge`](#)

new-real-network-host

Synopsis

new-real-network-host [“*name*”] “*interface*” [*persistent*]

Description

Creates an instantiated component of the class “real-network-host”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By
[real-network](#)

See Also[create-real-network-host](#)**new-real-network-router****Synopsis****new-real-network-router** [“*name*”] [“*interface*”] [“*gateway*”] “*ip*” “*netmask*”**Description**

Creates an instantiated component of the class “real-network-router”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By[real-network](#)**See Also**[create-real-network-router](#)**new-realtime****Synopsis****new-realtime** [“*name*”] *clock***Description**

In some cases simulated time may run faster than real time; this can happen if the OS is in a tight idle loop or an instruction halts execution waiting for an interrupt, or if the host machine is simply sufficiently fast. This can cause problems for programs that interact with the real world (for example the user), since time-outs may expire really fast.

A **realtime** object will, when enabled, periodically check the simulation speed and wait for a while if it is too high.

new-realtime creates a new **realtime** object.

Provided By[realtime](#)**See Also**[`<realtime>.enable`](#)**new-remote-frontend****Synopsis****new-remote-frontend** [*port*]

Description

Starts listening to incoming connection requests from remote debuggers. Simics will listen to TCP/IP requests on port specified by *port*, or 9123 by default. If *port* is given as zero, a random port will be selected.

The command returns the listening port.

Provided By

[frontend-server](#)

new-sample-gcache**Synopsis**

new-sample-gcache [“*name*”] [*assoc*] [*block_STC*] [*line_number*] [*line_size*] [*replacement_policy*] [*seed*] [*virtual_index*] [*virtual_tag*] [*write_allocate*] [*write_back*] [*penalty_read*] [*penalty_read_next*] [*penalty_write*] [*penalty_write_next*]

Description

Creates an instantiated component of the class “sample-gcache”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[timing-components](#)

See Also

[create-sample-gcache](#)

new-ser-link**Synopsis**

new-ser-link [“*name*”] [*buffer_size*] [*global_id*] [*goal_latency*]

Description

This command creates an instantiated component of the class **ser_link**.

The class description for the **ser_link** class: Serial link connecting two serial devices

***name* is Optional**

If not specified, the component will get a class-specific default name.

***buffer_size* is Optional**

The number of characters the link may buffer. Must be at least 1.

***global_id* is Optional**

Global identifier for use in distributed simulation or NIL if the link is not distributed.

***goal_latency* is Optional**

Goal latency in seconds for this link. See also the **set-min-latency** command.

Provided By
[ser-link](#)

new-signal-link

Synopsis

new-signal-link [“*name*”] [“*global_id*”] [*goal_latency*]

Description

This command creates an instantiated component of the class **signal_link**.

The class description for the **signal_link** class: The “signal_link” component represents a unidirectional signal that can be either high or low. It can be used to model electrical wires or more abstract binary signals.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***global_id* is Optional**

Global identifier for use in distributed simulation or NIL if the link is not distributed.

***goal_latency* is Optional**

Goal latency in seconds for this link. See also the **set-min-latency** command.

Provided By
[signal-link](#)

new-simple-memory-module

Synopsis

new-simple-memory-module [“*name*”] [*memory_megs*]

Description

This command creates an instantiated component of the class **simple_memory_module**.

The class description for the **simple_memory_module** class: A simple memory module.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***memory_megs* is Optional**

MB memory.

Provided By
[memory-comp](#)

new-std-etg

Synopsis

```
new-std-etg ["name"] ["mac_address"] "ip" "netmask" "dst_ip" ["gateway_ip"] [pps] [packet_size] [port]
```

Description

Creates an instantiated component of the class “std-etg”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

See Also

[create-std-etg](#)

new-std-ethernet-link**Synopsis**

```
new-std-ethernet-link ["name"] [latency] ["link_name"] [distributed] [frame_echo]
```

Description

Creates an instantiated component of the class “std-ethernet-link”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

See Also

[create-std-ethernet-link](#)

new-std-firewire-bus**Synopsis**

```
new-std-firewire-bus ["name"]
```

Description

Creates an instantiated component of the class “std-firewire-bus”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[firewire-components](#)

See Also

[create-std-firewire-bus](#)

new-std-firewire-sample-device

Synopsis

```
new-std-firewire-sample-device [“name”]
```

Description

Creates an instantiated component of the class “std-firewire-sample-device”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[firewire-components](#)

See Also

[create-std-firewire-sample-device](#)

new-std-generic-link**Synopsis**

```
new-std-generic-link [“name”] [latency] [“link_name”] [distributed]
```

Description

Creates an instantiated component of the class “std-generic-link”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

See Also

[create-std-generic-link](#)

new-std-graphics-console**Synopsis**

```
new-std-graphics-console [“name”] [window] [“title”]
```

Description

Creates an instantiated component of the class “std-graphics-console”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

See Also

[create-std-graphics-console](#)

new-std-host-serial-console

Synopsis

```
new-std-host-serial-console ["name"] ["port"]
```

Description

Creates an instantiated component of the class “std-host-serial-console”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

See Also

[create-std-host-serial-console](#)

new-std-mmc-card**Synopsis**

```
new-std-mmc-card ["name"] ["file"] size ["type"]
```

Description

This command creates an instantiated component of the class **std_mmc_card**.

The class description for the **std_mmc_card** class: The std_mmc_card component represents an MMC/SD/SDHC/SDIO card.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***file* is Optional**

File with disk contents for the full disk. Either a raw file or a CRAFF file

***size* is Required**

Card size, in bytes

***type* is Optional**

Card type ('mmc', 'sd', 'sdhc' or 'sdio'). Note that the card type will be adjusted by the model to handle large card sizes (i.e. card type will be forced to 'sdhc' if you create an 8 GB 'mmc' card).

Provided By

[std-components](#)

new-std-ms1553-link**Synopsis**

```
new-std-ms1553-link ["name"]
```

Description

Creates an instantiated component of the class “std-ms1553-link”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By[mil-std-1553-components](#)**See Also**[create-std-ms1553-link](#)**new-std-pcmcia-flash-disk****Synopsis****new-std-pcmcia-flash-disk** [“*name*”] *size* [“*file*”]**Description**

Creates an instantiated component of the class “**std-pcmcia-flash-disk**”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By[std-components](#)**See Also**[create-std-pcmcia-flash-disk](#)**new-std-sata-cdrom****Synopsis****new-std-sata-cdrom** [“*name*”]**Description**

This command creates an instantiated component of the class **std_sata_cdrom**.

The class description for the **std_sata_cdrom** class: The “**sata_cdrom**” component represents an Serial ATA CD-ROM.

***name* is Optional**

If not specified, the component will get a class-specific default name.

Provided By[std-components](#)**new-std-sata-disk****Synopsis****new-std-sata-disk** [“*name*”] [“*file*”] [*size*]**Description**

This command creates an instantiated component of the class **std_sata_disk**.

The class description for the **std_sata_disk** class: The “**sata_disk**” component represents a Serial ATA Disk.

***name* is Optional**

If not specified, the component will get a class-specific default name.

***file* is Optional**

File with disk contents for the full disk. Either a raw file or a CRAFF file.

***size* is Optional**

The size of the SATA disk in bytes. If it is omitted or less than the size of the image file, the disk size will be set to the same size as the image file.

Provided By

[std-components](#)

new-std-scsi-bus**Synopsis**

new-std-scsi-bus [“*name*”]

Description

Creates an instantiated component of the class “std-scsi-bus”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

See Also

[create-std-scsi-bus](#)

new-std-serial-link**Synopsis**

new-std-serial-link [“*name*”] [*latency*] [“*link_name*”] [*distributed*] [*throttle*]

Description

Creates an instantiated component of the class “std-serial-link”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

See Also

[create-std-serial-link](#)

new-std-server-console**Synopsis**

new-std-server-console [“*name*”] [*telnet_port*]

Description

Creates an instantiated component of the class “std-server-console”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

See Also

[create-std-server-console](#)

new-std-service-node**Synopsis**

new-std-service-node [“*name*”]

Description

Creates an instantiated component of the class “std-service-node”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[service-node](#)

See Also

[create-std-service-node](#)

new-std-telnet-console**Synopsis**

new-std-telnet-console [“*name*”] [*telnet_port*]

Description

Creates an instantiated component of the class “std-telnet-console”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

See Also

[create-std-telnet-console](#)

new-std-text-console**Synopsis**

new-std-text-console [“*name*”] [*window*] [“*title*”] [“*bg_color*”] [“*fg_color*”] [“*x11_font*”] [“*win32_font*”] [*width*] [*height*]

Description

Creates an instantiated component of the class “std-text-console”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

See Also

[create-std-text-console](#)

new-std-text-graphics-console**Synopsis**

```
new-std-text-graphics-console ["name"] [window] ["title"] ["bg_color"] ["fg_color"] ["x11_font"] ["win32_font"]
```

Description

Creates an instantiated component of the class “std-text-graphics-console”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[std-components](#)

See Also

[create-std-text-graphics-console](#)

new-symtable**Synopsis**

```
new-symtable ["name"] [file] [start] [-t] [-s] [-n]
```

Description

Creates a new symbol table, calling it *name* (or inventing a new name if omitted), and optionally loads debug info from *file*, using the Simics search path to do so.

If *start* is given, it is the (absolute) starting address of the file’s code. If *-t* is specified, *start* is interpreted as the address of the .text section (similarly to GDB); otherwise *start* is taken to be the address of the first executable segment.

If *-s* is given, only symbols are loaded and not any debug information.

If the current context of the selected CPU does not have an associated symbol table, it will be set to the newly created symbol table. This can be prevented with the *-n* flag, which means that the symbol table will not be automatically associated with any context.

Provided By

[symtable](#)

See Also

[`<symtable>.load-symbols, set-context`](#)

new-tcf-agent**Synopsis**

`new-tcf-agent ["name"] ["parameters"]`

Description

Command for creating a TCF agent in Simics. Not intended to be used directly.

Provided By

[`tcf-agent`](#)

See Also

[`debug-context`](#)

new-time-server**Synopsis**

`new-time-server [port] ["name"] [-poll]`

Description

Create a new time server that listens on port *port* (8123 by default); use *port* 0 for any port. Connected clients can query the virtual time, install alarm events that are triggered after a certain amount of virtual time, or install keepalive events that trigger at a periodic rate (real time). The name of the created time-server object is returned.

Provided By

[`time-server-c`](#)

new-tracer**Synopsis**

`new-tracer`

Description

Create a new tracer that connects to each CPU's memory space and traces instruction and data accesses.

Provided By

[`trace`](#)

See Also

[`<base-trace-mem-hier>.start`](#)

new-usb-disk

Synopsis

```
new-usb-disk ["name"] [size] ["file"]
```

Description

Creates an instantiated component of the class “usb-disk”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[usb-components](#)

See Also

[create-usb-disk](#)

new-usb-disk-from-image**Synopsis**

```
new-usb-disk-from-image image ["name"]
```

Description

Create a new USB disk object, with content from *file*. The image given in *file* should be a raw FAT file system image, not a craff file.

Provided By

[usb-components](#)

new-usb-hs-keyboard-comp**Synopsis**

```
new-usb-hs-keyboard-comp ["name"]
```

Description

This command creates an instantiated component of the class **usb_hs_keyboard_comp**. The class description for the **usb_hs_keyboard_comp** class: The High Speed USB Keyboard component class.

***name* is Optional**

If not specified, the component will get a class-specific default name.

Provided By

[usb-hid-components](#)

new-usb-keyboard-comp**Synopsis**

```
new-usb-keyboard-comp ["name"]
```

Description

This command creates an instantiated component of the class **usb_keyboard_comp**.

The class description for the **usb_keyboard_comp** class: The USB Keyboard component class.

***name* is Optional**

If not specified, the component will get a class-specific default name.

Provided By

[usb-hid-components](#)

new-usb-mouse-comp**Synopsis**

new-usb-mouse-comp [“*name*”]

Description

This command creates an instantiated component of the class **usb_mouse_comp**.

The class description for the **usb_mouse_comp** class: The USB Mouse component class.

***name* is Optional**

If not specified, the component will get a class-specific default name.

Provided By

[usb-hid-components](#)

new-usb-santa**Synopsis**

new-usb-santa [“*name*”]

Description

Creates an instantiated component of the class “usb-santa”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[usb-components](#)

See Also

[create-usb-santa](#)

new-usb-tablet-comp**Synopsis**

new-usb-tablet-comp [“*name*”]

Description

Creates an instantiated component of the class “usb-tablet-comp”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[usb-components](#)

See Also

[create-usb-tablet-comp](#)

new-usb-wacom-tablet-comp**Synopsis**

new-usb-wacom-tablet-comp [“*name*”]

Description

Creates an instantiated component of the class “usb-wacom-tablet-comp”. If *name* is not specified, the component will get a class-specific default name. The other arguments correspond to class attributes.

Provided By

[usb-components](#)

See Also

[create-usb-wacom-tablet-comp](#)

new-usb-xmas-tree**Synopsis**

new-usb-xmas-tree [“*name*”]

Description

This command creates an instantiated component of the class **usb_xmas_tree**.

The class description for the **usb_xmas_tree** class: The Xmas tree USB component.

***name* is Optional**

If not specified, the component will get a class-specific default name.

Provided By

[usb-comp](#)

new-wdb-remote**Synopsis**

new-wdb-remote [“*name*”] *processor* [*port*] *cpu-type* [*endianness*] *mem-size* *host-pool-base* *host-pool-size* [“*rt-name*”] [“*rt-version*”] [“*bootline*”] [*long-size*] [*pointer-size*] [*int-size*] [*rt-type*]

Description

Create a new WDB debug agent (an instance of the **wdb-remote** class).

The mandatory parameters are:

processor

This is the Simics object representing the processor in the system.

cpu-type

The type of CPU, as specified by the WDB protocol. Only used by the WDB client.

The command line tab completion system will suggest the allowed alternatives.

mem-size

The number of bytes of memory in the system. Only used by the WDB client.

host-pool-base, host-pool-size

The WDB protocol specifies that the host (Tornado or Workbench) owns a memory area on the target, which it may write whatever it wants to. These parameters specify the location and size of that area.

Note that **wdb-remote** cannot allocate this memory from the target OS, since it has no OS awareness. Thus, you will have to specify an area that the target OS will never use. Alternatively, if you do not plan to use the features that make use of this area (such as downloading code onto the target), you may give an arbitrary area. Using an area of size zero will not work, however.

The optional parameters are:

endianness

Specifies if the processor is little-endian or big-endian. If you do not specify it, it will be autodetected.

bootline

The system's boot line string. Only used by the WDB client.

rt-name

The name of the target OS, e.g. Linux or VxWorks. Only used by the WDB client. Default value is "Linux".

rt-version

The version of the target OS. Only used by the WDB client. Default value is "Linux 2.4".

rt-type

Valid values are "WDB_RT_NULL" corresponding to a standalone WDB agent, and "WDB_RT_VXWORKS", meaning a WDB agent integrated in VxWorks. Default value is "WDB_RT_NULL".

port

The UDP port the agent should listen on. Default is 0x4321 (17185).

long-size

Long size on target (in bytes), can be 2, 4, or 8. Only available in WDB5. Default is 4.

pointer-size

Pointer size on target (in bytes), can be 2, 4, or 8. Only available in WDB5. Default is 4.

int-size

Int size on target (in bytes), can be 2, 4, or 8. Only available in WDB5. Default is 4.

Provided By

[wdb-remote](#)

next-instruction**Alias**

nexti, ni

Synopsis

next-instruction

Description

next-instruction causes the simulation to run until it reaches another instruction, but will not stop in subroutine calls.

Provided By

[Simics Core](#)

See Also

[finish-function](#), [next-line](#), [reverse-next-instruction](#), [reverse-next-line](#), [reverse-step-instruction](#), [reverse-step-line](#), [step-instruction](#), [step-line](#), [uncall-function](#)

next-line**Alias**

next, n

Synopsis

next-line

Description

next-line causes the simulation to run until it reaches another source line, but will not stop in subroutine calls.

Provided By

[Simics Core](#)

See Also

[finish-function](#), [next-instruction](#), [reverse-next-instruction](#), [reverse-next-line](#), [reverse-step-instruction](#), [reverse-step-line](#), [step-instruction](#), [step-line](#), [uncall-function](#)

not**Synopsis****not** arg**Description**Returns TRUE if *arg* is false, and FALSE if not.**Provided By**[Simics Core](#)**object-exists****Synopsis****object-exists** "name"**Description**

Returns true if an object exists with the given name and false if not.

Provided By[Simics Core](#)**oct****Synopsis****oct** value [-u] [-p]**Description**Returns the parameter as a string in octal notation. This is similar to **print -o** *value*. To ignore any default digit grouping, the *-u* (unformatted) flag is used, while *-p* removes the radix prefix 0_o.**Provided By**[Simics Core](#)**See Also**[print](#), [hex](#), [bin](#), [dec](#), [digit-grouping](#)**or****Synopsis**arg1 **or** arg2**Description**Evaluates *arg1* and returns its value if it is true. Otherwise *arg2* is evaluated and its value is returned.**Provided By**[Simics Core](#)

output-file-start**Synopsis**

```
output-file-start filename [-overwrite] [-append]
```

Description

Start logging Simics output to the file *filename*. The file will not be overwritten unless the *-overwrite* flag is specified. The *-append* flag allows the output to be appended to the specified file.

The log file will contain regular Simics output and CLI error messages, as well as typed CLI commands, prefixed by the Simics prompt.

Provided By

[Simics Core](#)

See Also

[output-file-stop](#)

output-file-stop**Synopsis**

```
output-file-stop [filename]
```

Description

Stop sending output to the file *filename*. If no argument is given, then the command will disable all file output.

Provided By

[Simics Core](#)

See Also

[output-file-start](#)

output-radix**Synopsis**

```
output-radix [base] [group]
```

Description

Changes or displays the default output radix for numbers. It can be set to 2 for binary, 8 for octal, 10 for decimal, or 16 for hexadecimal output.

If *group* is non-zero, numbers will be grouped in groups of *group* digits, separated by underscores (_).

Currently, this affects the output of the **print**, **hex**, **dec**, **oct**, **bin** commands, and how return values from commands are displayed in CLI.

Without arguments, the current setting will be shown.

Provided By[Simics Core](#)**See Also**[digit-grouping](#), [print](#), [hex](#), [dec](#), [oct](#), [bin](#)**pcap-dump****Alias**

pcapdump

Synopsis

pcap-dump [*link*] [*probe*] [*device*] *filename*
pcap-dump-stop [*link*] [*probe*]

Description**Provided By**[eth-links](#)**pcap-dump-stop****Alias**

pcapdump-stop

Synopsis

pcap-dump-stop [*link*] [*probe*]
pcap-dump [*link*] [*probe*] [*device*] *filename*

Description**Provided By**[eth-links](#)**pdisable****Synopsis**

pdisable [*cpu-name|all*]
penable [*cpu-name|all*]

Description

Enables or disables a processor. If no processor is specified, then the current processor will be acted on. If the flag `-all` is passed, all processors will be enabled/disabled.

A disabled processor is simply stalled for an infinite amount of time.

Provided By
[Simics Core](#)

See Also
[pstatus](#)

penable

Synopsis

penable [*cpu-name|all*]
pdisable [*cpu-name|all*]

Description

Enables or disables a processor. If no processor is specified, then the current processor will be acted on. If the flag `-all` is passed, all processors will be enabled/disabled.

A disabled processor is simply stalled for an infinite amount of time.

Provided By
[Simics Core](#)

See Also
[pstatus](#)

pid

Synopsis
pid

Description

Return the process identity of the Simics process itself. Useful when attaching a remote debugger for example.

Provided By
[Simics Core](#)

pipe

Synopsis
pipe [-h] “*command*” “*pipe*”

Description

This command runs *command* at the Simics prompt and pipes the output (stdout) through the external program *pipe*’s stdin.

By default, commands run this way will be formatted to be machine readable rather than human readable. For example, this turns off multi-column output in commands such as **list-classes**. By specifying the `-h` flag, the output will be sent in human readable mode.

This command handles output and errors the same way as the **shell** command does.

Provided By
[Simics Core](#)

See Also
[shell](#)

popd

Synopsis
popd [-n]

Description

Pops a directory off the directory stack and, unless the `-n` option is specified, change current working directory to that directory.

Provided By
[Simics Core](#)

See Also
[dirs, pushd](#)

pos

Synopsis
pos (*line|function*)

Description

Finds the address of a source line or a function in a given file (e.g., `myfile.c:4711` or `myfile.c:myfunction`). If only a line number is specified, the command looks at the current value of the program counter to determine the current file. It may be necessary to put the argument inside double quotes for it to be parsed correctly.

Provided By
[Simics Core](#)

See Also
[stack-trace, psym, symval](#)

pow

Synopsis
arg1 pow arg2

Description

Return the `arg1` to the power of `arg2`.

Provided By
[Simics Core](#)

pregs**Synopsis****pregs** [*cpu-name*] [-all]**Description**

Prints the current integer register file of the processor *cpu-name*. If no CPU is specified, the current CPU will be selected. The **-all** flag causes additional registers, such as control registers and floating point registers to be printed.

Provided By[Simics Core](#)**print****Alias**

p

Synopsis**print** [-x|-o|-b|-s|-d] *value* [*size*]**Description**

Prints *value* in hexadecimal (-x), decimal (-d), octal (-o), or binary (-b) notation. Default is to use the notation specified by the **output-radix** command.

Use -s to convert the value to signed integers. *size* is the bit width to use. E.g., **print -x 257 8** will print 0x1. Valid sizes are 8, 16, 32, 64, and 128 bits. Default size is 64.

Provided By[Simics Core](#)**See Also**[output-radix](#)**print-event-queue****Alias**

peq

Synopsis**print-event-queue** [*obj*] [*queue*] [-i]**Description**

The simulator keeps an event queue for each processor. Interrupts, exceptions, and other events are posted on this event queue. For each event, the time to its execution and a brief description of it are printed. The time unit depends on the timing model of the queue; the default is the number of instructions before the event is triggered. If no CPU is specified, the selected frontend CPU is used. A *queue* argument of 0 means that only the step queue is printed, and for 1, only the time queue. Default is to print both queues. The flag **-i** enables printing of Simics internal events.

Provided By
[Simics Core](#)

print-time

Alias
 ptime

Synopsis
print-time [*cpu-name*] [-s] [-c] [-t] [-all]

Description

Prints the number of steps and cycles that a processor has executed. The cycle count is also displayed as simulated time in seconds.

If called from a processor namespace (e.g., **cpu0.print-time**), the time for that processor is printed. Otherwise, the time for the current processor is printed, or, if the **-all** flag is given, the time for all processors.

If the **-c** flag used, the cycle count for the processor is returned and nothing is printed. The **-s** flag is similar and returns the step count and **-t** returns the time in seconds as a floating point value.

A step is a completed instruction or an exception. An instruction that fails to complete because of an exception will count as a single step, including the exception.

Provided By
[Simics Core](#)

pselect

Alias
 psel

Synopsis
pselect [*cpu-name*]

Description

Use this command to select a default processor for the frontend. Many commands that have a processor as argument operates on the default processor when the argument is left out. Note that selecting processors does not affect in which order they are executed when multiprocessing is simulated.

Without any argument, this command will print the currently selected processor. It also returns the name of the currently selected processor, allowing scripting.

Provided By
[Simics Core](#)

pstatus**Synopsis****pstatus****Description**

Show the enabled/disabled status of all processors in the Simics session.

Provided By[Simics Core](#)**See Also**[penable](#)**psym****Synopsis****psym "expression"****Description**

Evaluates *expression* in the current stack frame, and prints the result in a human-readable form. The only C operators allowed are casts, indirection, member selection, and `sizeof` (thus no arithmetic). You may need to surround the *expression* by double quotes if it contains certain meta-characters.

Provided By[Simics Core](#)**See Also**[stack-trace](#), [frame](#), [symval](#)**pushd****Synopsis****pushd [-n] [path]****Description**

Pushes the directory *path* on top of the directory stack, or exchanges the topmost two directories on the stack. If *-n* is given, only change the contents of the stack, but do not change current working directory.

Provided By[Simics Core](#)**See Also**[dirs](#), [popd](#)**pwd**

Synopsis
pwd

Description

Return the working directory of Simics. Similar to the shell command ‘pwd’ (print working directory).

When used as part of an expression, returns the current working directory as a string.

Provided By

[Simics Core](#)

See Also

[cd](#), [ls](#)

python

Synopsis

python “*exp*”
‘*exp*’

Description

exp will be evaluated in the Python environment and the result returned to the frontend.

For example:

```
print -x 'SIM_step_count(SIM_current_processor())'
$all_objects = python "SIM_get_all_objects()"
```

Both expressions and statements can be run, but for statements the @ command can be used instead.

Provided By

[Simics Core](#)

See Also

[@](#), [run-python-file](#)

quit

Alias

q, exit

Synopsis

quit [*status*] [-f] [-d]

Description

Exit Simics gracefully. The optional *status* argument is the exit status of Simics. When issued in a secondary command-line, such as the telnet-frontend, the *f* force flag must be given to exit Simics while the *-d* can be used to disconnect the command line only keeping Simics running. Disconnecting the command line can also be done with Ctrl-D

Provided By
[Simics Core](#)

range**Synopsis****range** *start* [*end*] [*step*]**Description**

Return a list of integers from *start* up to *end* - 1. A single argument is interpreted as *end* with 0 as *start*. The optional *step* specifies the increment and may be negative.

Provided By
[Simics Core](#)

read-configuration**Synopsis****read-configuration** *file* [“*prefix*”]**Description**

Read configuration from *file*. The configuration can be either a checkpoint or an initial configuration. For information about how to create or modify configurations, see the *Advanced Features of Simics* manual. The *prefix* argument can be used to add a name prefix to all objects loaded from the checkpoint file.

Provided By
[Simics Core](#)

See Also[write-configuration](#)**read-reg****Synopsis**

```
read-reg [cpu-name] “reg-name”
<int_register>.read-reg “reg-name”
```

Description

This command reads a register from an object implementing the `int_register` interface. For example, to read the `eax` register in an x86 processor called `cpu0`, write **read-reg** `cpu0 eax`. You can also use the method variant: `cpu0.read-reg eax`, or the more convenient variant `%eax` that reads a register from the selected frontend CPU.

If no *cpu-name* is supplied, the current frontend processor is used.

Provided By
[Simics Core](#)

See Also

[%](#), [write-reg](#), [pregs](#), [pselect](#)

read-variable**Synopsis**

read-variable “*variable*” [*alt*]

Description

Returns the value of the CLI variable named *variable* if it is defined. If the variable does not exist, then *alt* is returned or FALSE if *alt* is not specified.

Note that `read-variable foo` returns the value of variable `$foo`, whereas `read-variable $bar` returns the value of the variable whose name is stored in `$bar`.

Provided By

[Simics Core](#)

See Also

[defined](#), [\\$](#)

readme**Synopsis**

readme

Description

Prints various useful information (README) about Simics.

Provided By

[Simics Core](#)

resolve-file**Alias**

`lookup-file`

Synopsis

resolve-file “*filename*” [-query]

Description

Looks for the file or directory *filename* in the Simics search path, starting with the current working directory. If it is found, its complete path is returned. If not found and if *-query* is used the boolean value FALSE is returned.

Warning: In the next major Simics release, this command will raise an error if the file is not found, unless the *-query* flag is used. Currently FALSE is returned together with a deprecation warning.

Provided By
[Simics Core](#)

See Also
[file-exists](#)

restart-simics

Synopsis
restart-simics [*file*] [-fast] [-stall]

Description

Exits the current Simics session and starts a new empty one. The optional *file* argument can be the name of a Simics script or checkpoint file to open after restarting. The *-fast* and *-stall* flags can be used to override the default CPU mode for the new session.

Provided By
[Simics Core](#)

reverse

Alias
 rev

Synopsis
reverse [*count*]

Description

Runs the simulation in reverse. The simulation will stop at any breakpoints or after at most *count* instructions (if specified).

Before the machine can be reversed, at least one time bookmark has to be set. Reverse operations are possible in the region following the first (oldest) time bookmark.

Provided By
[Simics Core](#)

See Also
[set-bookmark](#), [skip-to](#), [reverse-to](#)

reverse-cycles

Synopsis
reverse-cycles [*count*]

Description

Runs the simulation in reverse. The simulation will stop at any breakpoints or after at most *count* cycles (if specified).

Before the machine can be reversed, at least one time bookmark has to be set. Reverse operations are possible in the region following the first (oldest) time bookmark.

Provided By[Simics Core](#)**See Also**[set-bookmark](#), [skip-to](#), [reverse-to](#)**reverse-next-instruction****Alias**

rnexti, rni

Synopsis**reverse-next-instruction****Description**

reverse-next-instruction causes the simulation to run backwards until it reaches another instruction, but will not stop in subroutine calls.

Provided By[Simics Core](#)**See Also**[finish-function](#), [next-instruction](#), [next-line](#), [reverse-next-line](#), [reverse-step-instruction](#), [reverse-step-line](#), [step-instruction](#), [step-line](#), [uncall-function](#)**reverse-next-line****Alias**

rnext, rn

Synopsis**reverse-next-line****Description**

reverse-next-line causes the simulation to run backwards until it reaches another source line, but will not stop in subroutine calls.

Provided By[Simics Core](#)**See Also**[finish-function](#), [next-instruction](#), [next-line](#), [reverse-next-instruction](#), [reverse-step-instruction](#), [reverse-step-line](#), [step-instruction](#), [step-line](#), [uncall-function](#)**reverse-step-instruction****Alias**

unstep-instruction, rstepi

Synopsis**reverse-step-instruction** [*count*]**Description**

Executes *count* instructions in reverse, printing each instruction at each step. *count* defaults to one.

The **reverse-next-instruction** command is similar except that it does not reverse into called functions.

Provided By

[Simics Core](#)

See Also

[reverse-next-instruction](#), [reverse-next-line](#), [uncall-function](#), [reverse-step-line](#)

reverse-step-line**Alias**

rstep, rs

Synopsis**reverse-step-line****Description**

reverse-step-line causes the simulation to run in reverse until it reaches another source line.

Provided By

[Simics Core](#)

See Also

[finish-function](#), [next-instruction](#), [next-line](#), [reverse-next-instruction](#), [reverse-next-line](#), [reverse-step-instruction](#), [step-instruction](#), [step-line](#), [uncall-function](#)

reverse-to**Alias**

revto

Synopsis**reverse-to** ("bookmark" | *step|cycle*)**Description**

Runs the simulation in reverse. The simulation will stop at any breakpoints or at the specified (absolute) point in time if no breakpoints occurred.

At least one time bookmark has to be set before any reverse operations are possible.

Provided By[Simics Core](#)**See Also**[set-bookmark](#), [reverse](#), [skip-to](#)**rexec-limit****Alias**

rlimit

Synopsis**rexec-limit** [*size_mb*] [*steps*]**Description**

Tunes various overall resource limits for reverse execution.

size_mb limits the amount of memory used for reverse execution. If the value is set to 0, no limit is imposed.

steps limits the scope of reversibility to the specified number of steps. If the value is set to 0, no limit is imposed.

Provided By[Simics Core](#)**run****Alias**

continue, c, r

Synopsis**run** [*count*]**Description**

Starts or continues executing instructions. If a *count* argument is provided, Simics will execute *count* number of instructions and stop.

Provided By[Simics Core](#)**See Also**[step-instruction](#), [run-cycles](#)**run-command-file****Synopsis****run-command-file** *file* [-local]

Description

This command starts executing a Simics script. A Simics script is an ordinary text file that contains Simics commands. One command on each line. The syntax used is exactly the same as when commands are typed at the Simics prompt. The # character is used as the start of a comment and applies to the rest of the line.

If the *-local* flag is supplied, the script will run with its own copy of all global CLI variables. When the script has finished executing, the previous variable set is restored.

Python code can also be executed by prefixing the line with @. Multi-line Python statements can be used by leaving a blank line at the end of the statement. Only the first line should have an @ in this case.

Simics scripts usually ends with the suffix ".simics" but this is only a convention. The suffix is ignored by Simics.

This is an example of a Simics script:

```
# This is a Simics script

break 0xfffc000 # set a breakpoint
run
echo "breakpoint reached"
run-command-file another-script.simics
```

Simics scripts can be executed directly when Simics has started by using the *-x* command line option.

If a command fails or the user presses Ctrl-C, the Simics script is interrupted and control returns to the Simics prompt.

run-command-file uses Simics's Search Path and path markers (%simics%, %script%) to find the script to run. Refer to *The Command Line Interface* chapter of the *Hindsight User's Guide* manual for more information.

Provided By

[Simics Core](#)

See Also

[run-python-file](#), [add-directory](#)

run-cycles

Alias

continue-cycles, cc, rc

Synopsis

run-cycles [*count*]

Description

Starts or continues executing instructions. If a *count* argument is provided, Simics

will execute *count* number of cycles and stop. Note that running *count* cycles may or may not be equivalent to running *count* instructions depending on the way Simics is configured. Refer to the *Understanding Simics Timing* application note.

Provided By

[Simics Core](#)

See Also

[step-cycle](#), [run](#)

run-python-file

Synopsis

run-python-file *filename*

Description

Read Python code from *filename*. Any definitions are entered into the top level namespace in the Python environment. Uses the Simics search path to locate *filename*. This command can be used to start Python scripts inside Simics.

run-python-file uses Simics's Search Path and path markers (%simics%, %script%) to find the script to run. Refer to *The Command Line Interface* chapter of the *Hindsight User's Guide* manual for more information.

Provided By

[Simics Core](#)

See Also

[python](#), [@](#), [run-command-file](#), [add-directory](#)

run-seconds

Alias

continue-seconds

Synopsis

run-seconds *seconds*

Description

Starts or continues executing instructions for a period of *seconds* of virtual time and stops the simulation.

Provided By

[Simics Core](#)

See Also

[run-cycles](#)

save-component-template**Synopsis****save-component-template** *file***Description**

Save a configuration file with only component objects and their connection information. This template corresponds to an empty machine configuration, without any software setup. The saved component template can be loaded into Simics using the **read-configuration** command, producing a collection of non-instantiated components.

Provided By[Simics Core](#)**See Also**[read-configuration](#), [write-configuration](#), [list-components](#)**save-persistent-state****Synopsis****save-persistent-state** *file* [-z] [-u] ["*comment*"]**Description**

Save the persistent simulator state to *file*. Persistent data typically includes disk images, NVRAM and flash memory contents and clock settings, i.e. data that survive reboots. The persistent state is saved as a standard Simics configuration.

Use the *-z* flag for compressed images, or *-u* for uncompressed. The default is taken from the preference object.

To add a description to the persistent state, use the *comment* argument. The comment is saved in the *info* file in the persistent state bundle.

Provided By[Simics Core](#)**See Also**[load-persistent-state](#), [write-configuration](#)**save-preferences****Synopsis****save-preferences****Description**

Save the current user preference settings. The preferences will be loaded automatically the next time Simics is started.

Provided By[Simics Core](#)

See Also[list-preferences](#)**script-branch****Synopsis****script-branch { *commands* }****Description**

Starts a block of commands as a separate branch. The **wait-for-** commands can be used to postpone the execution of a script branch until a selection action occurs.

Provided By[Simics Core](#)**See Also**

[list-script-branches](#), [interrupt-script-branch](#), [wait-for-script-barrier](#), [wait-for-script-pipe](#), [wait-for-breakpoint](#), [<text-console>.wait-for-string](#), [<int_register>.wait-for-register-read](#), [<int_register>.wait-for-register-write](#), [<cycle>.wait-for-cycle](#), [<step>.wait-for-step](#), [<cycle>.wait-for-time](#), [<os-awareness>.wait-for-activated](#), [<os-awareness>.wait-for-deactivated](#)

script-pipe-has-data**Synopsis****script-pipe-has-data** *pipe***Description**

Returns true if the script pipe has data that can be read and false if it is empty.

Provided By[Simics Core](#)**See Also**[script-branch](#), [create-script-pipe](#), [wait-for-script-pipe](#), [add-data-to-script-pipe](#)**select-profiles****Synopsis****select-profiles** [*profiler* ...] [*view*] [-add]**Description**

Specifies what profiles, if any, should be shown in subsequent source code listings, and in what order. Use without argument to show none.

Provided By[symtable](#)

set**Synopsis**

```
set address value [size] [-l] [-b]
<memory-space>.set address value [size] [-l] [-b]
<memory-space>.write address value [size] [-l] [-b]
<port-space>.set address value [size] [-l] [-b]
<port-space>.write address value [size] [-l] [-b]
```

Description

Set the *size* bytes of physical memory at location *address* to *value*. The default *size* is 4 bytes, but can be anywhere between 1 and 8 (inclusive) for memory-spaces and between 1 and 4 (inclusive) for port-spaces.

If *value* is larger than the specified size, behavior is undefined.

The *-l* and *-b* flags are used to select little-endian and big-endian byte order, respectively. If neither is given, the byte order of the currently selected processor is used.

The **set** command performs the access in inquiry mode without triggering any side-effects while **write** may trigger side-effects.

If the *value* argument is a list, then each item is written to memory as a value of *size* bytes, starting at *address*. The byte order flags operate on each item in the list.

The non-namespace version of this command operates on the physical memory associated with the current processor.

Provided By

[Simics Core](#)

See Also

[get](#), [x](#), [pselect](#)

set-bookmark**Alias**

bookmark

Synopsis

```
set-bookmark ["bookmark"]
```

Description

Associates a time bookmark with the current point in the simulation. The bookmark can be used as a target to the **skip-to** command.

Reverse operations are possible in the region following the first (i.e. oldest) time bookmark.

Setting a time bookmark can cause a certain reduction in forward simulation performance (deleting all bookmarks will restore the original performance).

The name of the created bookmark is returned.

Provided By[Simics Core](#)**See Also**[delete-bookmark](#), [reverse](#), [skip-to](#)**set-component-prefix — deprecated****Synopsis****set-component-prefix** “*prefix*”**Description**

This command is deprecated; use [hierarchical components](#) instead.

Sets a string prefix that will be added to the names of all component objects that are created after the invocation of this command.

Provided By[Simics Core](#)**See Also**[get-component-prefix](#)**set-context****Synopsis****set-context** “*context*”**Description**

Sets the current context of a processor to *context*. If the context does not exist, it is created.

Provided By[Simics Core](#)**See Also**[new-context](#), <*context*>.symtable**set-memory-limit****Synopsis****set-memory-limit** [*limit*] [*swapdir*]**Description**

Limits the in-memory footprint of all image objects to *limit* megabytes. This only limits the memory consumed by image pages in memory. While this is typically a very large part of Simics’s memory usage, other data structures are not limited by this command.

If *limit* is zero, the memory limit is removed. If *swapdir* is specified, it indicates a directory to use for swap files. If no argument is given, the current setting is displayed. Simics sets a default memory limit at startup that depends on the amount of memory, and number of processors, on the host system.

Provided By

[Simics Core](#)

set-min-latency

Synopsis

set-min-latency *min-latency*

Description

Set the min-latency, in seconds, of the **default_sync_domain** object (creating it if it does not exist).

Provided By

[Simics Core](#)

set-pattern

Synopsis

set-pattern *id* “*pattern*” “*mask*”

Description

When set for breakpoint *id* Simics will only break on instructions with a certain bit-pattern. First the *mask* will be applied to the instruction and then the result will be compared with the *pattern*. For example **set-pattern** 1 “0x0100” “0x0101” will specialize breakpoint 1 to break on instructions whose first byte has the lowest bit set and the second not.

Note that pattern and mask are supplied as strings with string byte order (low address first).

Set pattern and mask to the empty string (“”) to remove this extra condition.

Provided By

[Simics Core](#)

Note

Only supported for execution breakpoints.

See Also

[set-prefix](#), [set-substr](#)

set-pc

Synopsis

set-*pc* *address*

Description

Set program counter (instruction pointer) of the processor (defaults to the current frontend processor) to *address*.

Provided By

[Simics Core](#)

set-prefix**Synopsis**

set-prefix *id* “*prefix*”

Description

Set a syntax prefix for a breakpoint. When set Simics will only break on instructions with a certain syntax prefix. For example **set-prefix** 1 “add” will cause breakpoint 1 only to stop if the instruction begins with “add”. The text to compare the prefix with for an instruction is the one which the instruction is disassembled to. The comparison is case insensitive.

Set prefix to the empty string (“”) to remove this extra condition.

Provided By

[Simics Core](#)

Note

Only supported for execution breakpoints.

See Also

[set-substr](#), [set-pattern](#)

set-substr**Synopsis**

set-substr *id* “*substr*”

Description

When set Simics will only break on instructions with a certain syntax substring. For example **set-substr** 1 “r31” will make breakpoint 1 only stop if the instruction has a substring “r31”. The match is case insensitive.

Set sub-string to the empty string (“”) to remove this extra condition.

Provided By

[Simics Core](#)

Note

Only supported for execution breakpoints.

See Also

[set-prefix](#), [set-pattern](#)

set-thread-limit**Synopsis**

set-thread-limit [*limit*]

Description

Limits the number of threads Simics may use for multithreaded simulation. If *limit* is zero, the thread limit is removed.

If no argument is given, the current setting is displayed.

Provided By

[Simics Core](#)

shell**Synopsis**

shell “*exp*”

Description

Executes the *exp* argument in the system command line interpreter. For Unix, this is the default shell. For Windows, this is cmd.exe.

If the command returns a non-zero exit status, a command line error will be signalled. The actual exit status number cannot be obtained from this command.

When used non-interactively, any messages printed to standard output will be returned by the **shell** command:

\$today = (shell “date +%F”) on Unix; or
\$today = (shell “date /t”) on Windows.

Provided By

[Simics Core](#)

See Also

[!](#), [pipe](#), [try](#)

show-memorymap**Synopsis**

show-memorymap

Description

Show the current memory map used by the debugger. This is what is built with the add-symbol-file command. The memory map is used in addition to the memory map configured in Eclipse.

Provided By
[tcf-agent](#)

See Also

[add-symbol-file](#), [clear-memorymap](#)

show-pathmap

Synopsis
show-pathmap

Description

Show all entries in the pathmap.

The pathmap is the name for the translations from paths in target binaries to paths on the Simics host. This pathmap is specific to the command line interface to the debugger, it is not shared with the Eclipse frontend to the debugger.

Provided By
[tcf-agent](#)

See Also

[add-pathmap-entry](#)

signed

Synopsis
signed *int*
signed16 *int*
signed32 *int*
signed64 *int*
signed8 *int*

Description

Interpret an integer, *int*, as a signed value of a specific bit width. For example **signed16** **0xffff** will return -1. The **signed** command assumes a 64 bit width.

Provided By
[Simics Core](#)

signed16

Synopsis
signed16 *int*
signed *int*
signed32 *int*
signed64 *int*
signed8 *int*

Description

Interpret an integer, *int*, as a signed value of a specific bit width. For example **signed16** 0xffff will return -1. The **signed** command assumes a 64 bit width.

Provided By

[Simics Core](#)

signed32**Synopsis**

- signed32** *int*
- signed** *int*
- signed16** *int*
- signed64** *int*
- signed8** *int*

Description

Interpret an integer, *int*, as a signed value of a specific bit width. For example **signed16** 0xffff will return -1. The **signed** command assumes a 64 bit width.

Provided By

[Simics Core](#)

signed64**Synopsis**

- signed64** *int*
- signed** *int*
- signed16** *int*
- signed32** *int*
- signed8** *int*

Description

Interpret an integer, *int*, as a signed value of a specific bit width. For example **signed16** 0xffff will return -1. The **signed** command assumes a 64 bit width.

Provided By

[Simics Core](#)

signed8**Synopsis**

- signed8** *int*
- signed** *int*
- signed16** *int*
- signed32** *int*
- signed64** *int*

Description

Interpret an integer, *int*, as a signed value of a specific bit width. For example **signed16** 0xffff will return -1. The **signed** command assumes a 64 bit width.

Provided By

[Simics Core](#)

simulation-replaying**Synopsis**

simulation-replaying

Description

Returns true if the simulation is currently replaying, i.e. running forward again where it has executed before after having reversed previously. Returns false if the simulation is reversing, running forward for the first time or if it is stopped. The **clear-recorder** command will tell Simics to forget that it is replaying.

Provided By

[Simics Core](#)

See Also

[run](#), [stop](#), [clear-recorder](#), [simulation-running](#), [simulation-reversing](#)

simulation-reversing**Synopsis**

simulation-reversing

Description

Returns true if the simulation is currently reversing and false if it is running forward or is stopped.

Provided By

[Simics Core](#)

See Also

[run](#), [stop](#), [simulation-running](#), [simulation-replaying](#)

simulation-running**Synopsis**

simulation-running

Description

Returns true if the simulation is currently running (in any direction) and false if it is stopped.

Provided By
[Simics Core](#)

See Also
[run](#), [stop](#), [simulation-reversing](#), [simulation-replaying](#)

skip-to

Synopsis
skip-to (“*bookmark*”|*step*|*cycle*)

Description

Skips to the specified point in time. The target position can either be a *label* previously defined by **set-bookmark** or a step or cycle count.

Provided By
[Simics Core](#)

split-string

Synopsis
split-string [-type] “*separator*” “*string*” [-str]

Description

Split a string using a supplied separator or based on its type. When the **-type** flag is supplied, *separator* has to be one of the pre-defined string types **ethernet**, **ipv4**, **ipv6**, **filename** or **path**. The return value from the command is a list of strings or integers depending on the type. The **-str** can be used to always get a return value of strings.

Supported conversion types:

ethernet

Split Ethernet MAC address into list of six integers.

ipv4

Split IPv4 address in dot-decimal notation into list of four integers.

ipv6

Split IPv6 address in hexadecimal, colon-separated notation into list of eight integers.

filename

Split a filesystem path into a directory part and a filename part.

path

Split a filesystem path into one part for each directory level.

The filename and path types work on both Unix and Windows style filesystem paths independent of the host system.

Examples using pre-defined string types:

```
"ethernet" "ff:fe:01:55:88:11" → [255, 254, 1, 85, 136, 17]
"ipv4" "255.0.0.1" → [255, 0, 0, 1]
"ipv6" "2001:db8:85a3::7334" → [8193, 3512, 34211, 0, 0, 0, 0, 29492]

"ipv6" "::1" → [0,0,0,0,0,0,0,1]
"ipv6" -str "1:fffff::1" → ["1","fffff","0","0","0","0","0","1"]
"filename" "/" → ["/", ""]
"filename" "/home/user/x" → ["/home/user", "x"]
"filename" "\\home\user\x" → ["\\home\user", "x"]
"filename" "c:\x" → ["c:\", "x"]
"path" "c:\home\user\x" → ["c:\", "home", "user", "x"]
```

Provided By
[Simics Core](#)

stack-trace

Alias

bt, where

Synopsis

stack-trace [*maxdepth*]

Description

Displays a stack trace in the current context of the specified processor, or the current processor if none was specified. At most *maxdepth* frames are shown, 64 by default.

Provided By
[Simics Core](#)

See Also

[frame](#)

state-assertion-connect

Synopsis

state-assertion-connect [“*server*”] [*port*] [“*compression*”] [*align*] [*post_events*] [“*name*”]

Description

This command connects to a state-assertion receiver so that all data gathered during the state recording will be sent over to the receiver.

- *server* receiver host waiting for the connection

- *port* port number on which the receiver is waiting for a connection
- *compression* is the compression used (none, gz)
- *name* is the name of the object to be created. Default is saX where X is a number.

Provided By[state-assertion](#)**state-assertion-create-file****Synopsis****state-assertion-create-file** *file* [“*compression*”] [“*name*”] [*align*] [*post_events*]**Description**

This command creates a state assertion file.

- *file* is the name of the file to be created
- *compression* is the compression used (none, gz)
- *name* is the name of the object to be created. Default is saX where X is a number.
- *align* is the alignment of the structures inside the file. It can be useful to set it so that objects saving their state are sure to get correctly aligned structures. Default is 8 which is sufficient for most hosts.
- *post_events* tells state-assertion to post events by itself for recording and comparing. Default is true.

Provided By[state-assertion](#)**state-assertion-open-file****Synopsis****state-assertion-open-file** *file* [“*compression*”] [“*name*”] [*post_events*]**Description**

Open a state assertion file to compare it to the current execution.

- *name* is the name of the object. A default name in saX is provided if none is given.
- *file* is the name of the state assertion file
- *compression* is the compression used on the file (none, gz) - *post_events*

Provided By[state-assertion](#)**state-assertion-receive****Synopsis****state-assertion-receive** [*port*] [“*compression*”] [“*name*”] [*post_events*]

Description

Wait for a connection (state-assertion-connect) from a sender. The data received from the sender will be compared against the current execution.

- *port* indicates where simics should wait for the connection
- *compression* is the compression used on the file (none, gz) - *name* is the name of the object. A default name in saX is provided if none is given.

Provided By

[state-assertion](#)

state-assertion-simple-assert**Synopsis**

state-assertion-simple-assert [*file*] [“*compression*”] [*post_event*]

Description

This command asserts the current run against the file. You just have to run ‘c’ afterwards to begin the assertion process.

Provided By

[state-assertion](#)

state-assertion-simple-record**Synopsis**

state-assertion-simple-record [*file*] [“*compression*”] *object* [*steps*] [*type*]

Description

Create a file (by default /tmp/state-assertion-\$USER.gz) and save the state of *object* every *steps* steps. You just have to run ‘c’ afterwards to begin the recording.

object is the simics object whose state will be recorded. *steps* is the number of steps between each state recording (default is 1).

Provided By

[state-assertion](#)

stc-status**Synopsis**

stc-status
dstc-disable
dstc-enable
iostc-disable
iostc-enable
istc-disable
istc-enable

Description

These commands are for advanced usage only. They allow the user to control the usage of Simics-internal caches.

The Simulator Translation Caches (STCs) are designed to increase execution performance. The D-STC caches data translations (logical to physical to real (host) address). The I-STC caches instruction translations of taken jumps. Finally, the IO-STC caches logical to physical address translation for device accesses.

By default the STCs are **on**.

When a memory hierarchy is connected (such as a cache module) it must have been designed to work along with the STCs, or it may not be called for all the memory transactions it is interested in. These commands can be used to detect if too many translations are kept in the STCs, causing the simulation to be incorrect.

Turning the STCs off means that current contents will be flushed and no more entries will be inserted into the STCs.

Provided By

[Simics Core](#)

step-break**Alias**

sb, sim-break

Synopsis

step-break [*cpu-name*] *instructions*

Description

Sets a breakpoint so that the CPU will stop after executing *instructions* number of steps from the time the command was issued. If the CPU is not specified the selected frontend processor will be used (see **pselect**).

To list all breakpoints set use the command **list-breakpoints**.

Provided By

[Simics Core](#)

See Also

[step-break-absolute](#), [cycle-break](#), [cycle-break-absolute](#), [list-breakpoints](#)

step-break-absolute**Alias**

sba, sim-break-absolute

Synopsis

step-break-absolute [*cpu-name*] *instructions*

Description

Set a breakpoint so that the selected CPU will stop after its step counter has reached the *instructions* value. If the CPU is not specified the selected frontend processor will be used (see [pselect](#)).

To list all breakpoints set use the command [list-breakpoints](#).

Provided By

[Simics Core](#)

See Also

[step-break-absolute](#), [cycle-break](#), [cycle-break-absolute](#), [list-breakpoints](#)

step-cycle**Alias**

sc

Synopsis

step-cycle [*count*]

Description

Executes *count* cycles, printing the next instruction to be executed at each cycle. *count* defaults to one.

Provided By

[Simics Core](#)

See Also

[run](#), [step-instruction](#)

step-instruction**Alias**

si, stepi

Synopsis

step-instruction [*count*] [-r]

Description

Executes *count* instructions, printing the next instruction to be executed at each step. *count* defaults to one. With the -r flag, register changes will also be printed.

Provided By

[Simics Core](#)

See Also

[run](#), [step-cycle](#)

step-line**Alias**

step, s

Synopsis**step-line****Description****step-line** causes the simulation to run until it reaches another source line.**Provided By**[Simics Core](#)**See Also**[finish-function](#), [next-instruction](#), [next-line](#), [reverse-next-instruction](#), [reverse-next-line](#), [reverse-step-instruction](#), [reverse-step-line](#), [step-instruction](#), [uncall-function](#)**stop****Synopsis****stop [-a] ["message"]****Description**

Stops the simulation as soon as possible. If the –a argument is given, any command script running will also be interrupted. A *message* to be printed on the console when the simulation stops can also be supplied. If the **stop** command is called from a script branch, then the branch will wait for the simulation to finish before the command returns. When called from the main branch, the simulation may still execute when the command returns.

Provided By[Simics Core](#)**See Also**[run](#)**sym-address****Synopsis****sym-address (line|“expression”)****Description**

Returns the address of the evaluated *expression*, for example a symbol, in the current stack frame or the source reference in the file:line or line format. The argument may have to be surrounded by double quotes if it contains certain meta-characters.

Provided By[Simics Core](#)

See Also

[sym-value](#), [sym-type](#), [sym-string](#), [sym-source](#)

sym-file**Synopsis**

sym-file ("function" | address)

Description

Returns the source file with complete path in the host machine's file system for for *address* or *function*.

Provided By

[Simics Core](#)

See Also

[sym-function](#), [sym-line](#), [sym-source](#), [sym-address](#)

sym-function**Synopsis**

sym-function *address*

Description

Returns the function at the specified *address* in memory.

Provided By

[Simics Core](#)

See Also

[sym-file](#), [sym-line](#), [sym-source](#), [sym-address](#)

sym-line**Synopsis**

sym-line ("function" | address)

Description

Returns the source line for for *address* or *function*.

Provided By

[Simics Core](#)

See Also

[sym-function](#), [sym-file](#), [sym-source](#), [sym-address](#)

sym-source

Synopsis

sym-source ("function"|"address)

Description

Prints the source file, line and function for *address* or *function*..

Provided By

[Simics Core](#)

See Also

[sym-file](#), [sym-line](#), [sym-function](#), [sym-address](#)

sym-string**Synopsis**

sym-string ("expression"|"address")

Description

Interprets *address* or the value of *expression* as a pointer to a string in target memory, returning the string. The expression, if used, is evaluated in the current stack frame. The argument may have to be surrounded by double quotes if it contains certain meta-characters.

Provided By

[Simics Core](#)

See Also

[sym-value](#), [sym-type](#), [sym-address](#)

sym-type**Synopsis**

sym-type "expression"

Description

Returns the type of the symbol, or the evaluated *expression*, in the current stack frame. The argument may have to be surrounded by double quotes if it contains certain meta-characters.

Provided By

[Simics Core](#)

See Also

[sym-value](#), [sym-address](#), [sym-string](#)

sym-value

Synopsis

sym-value “*expression*”

Description

Evaluates *expression* in the current stack frame. The argument may have to be surrounded by double quotes if it contains certain meta-characters. When **sym-value** is used in a CLI expression, the value of the supplied expression is returned. When used stand-alone, the value is pretty-printed.

Provided By

[Simics Core](#)

See Also

[stack-trace](#), [frame](#), [sym-type](#), [sym-address](#), [sym-string](#)

symval**Alias**

sym

Synopsis

symval “*expression*”

Description

Evaluates *expression* in the current stack frame. The only C operators allowed are casts, indirection, member selection, and `sizeof` (thus no arithmetic). You may need to surround the *expression* by double quotes if it contains certain meta-characters. In contrast to **psym**, the result is returned as an unadorned value that can be used in CLI expressions.

Provided By

[Simics Core](#)

See Also

[stack-trace](#), [frame](#), [psym](#)

sync-info**Synopsis**

sync-info

Description

Print the synchronization tree.

Provided By

[Simics Core](#)

system-info

Synopsis

```
system-info [file]
```

Description

Show information about both the host system and the target system. Includes information useful when analyzing correctness and performance problems.

Provided By

[Simics Core](#)

system-perfmeter**Synopsis**

```
system-perfmeter [sample_time] [-deactivate] [-realtime] [-cpu-idle] [-cpu-exec-mode]
[-cpu-host-ticks] [-summary] [-module-profile] [-window] [-top] [-disabled] [-mips]
[-emips] [-mem] [-shared] [-mips-win] [-no-log] [file]
```

Description

Activates performance measurement on a system running within Simics. The resulting printouts gives an idea on how fast Simics executes compared to the system being simulated.

How frequent the measurements should be presented is controlled with the *sample_time* parameter which represent the system time (virtual seconds) that should elapse for each sample, default is one second. Using the *-realtime* flag causes each sample to be in real time instead of virtual time.

To disable this feature use the *-deactivate* flag.

The *-top* flag opens a separate text window displaying some statistics on the execution, similar to the Unix top utility.

Similar, the *-mips-win* opens a window displaying only the current MIPS value which can be useful for demonstration.

The *-summary* causes a summary report to be printed out each time simulation is stopped. In the summary report includes the same kind of fields that you get for each sample, but the numbers are calculated based on the whole run, not just a sample, since the command was issued last time. That is, changing any flags or options to the command also resets the totals.

Specifying *-module-profile* enables profiling of the simulator.

Normally a text line with results is written as an output each measured sample. The *-window* flag opens a separate text window where the continued output is written instead of printing this in the Simics console.

The console printouts can be sent to a file specified by the *file* argument. The default is to output the result to the Simics console. If the *-window* flag is used together with a specified *file*, the output is written both to the file and to the separate window.

If no output is wanted at all, the *-no-log* flag can be used (can be useful when running with only *-top* or *-mips-win*).

All of the below flags are used to add various counters to the sample.

The *-mips* flag appends some MIPS values indicating how many million instruction per real second Simics has executed. The MIPS number printed is the number of instruction executed, including idle instructions. To see the MIPS value without the idle instructions (where only the instructions that are really executed in Simics are counted) you can use the *-emips*.

In some configurations processors might be disabled at start and started later by software. To see how many of the processors that are disabled each sample use the *-disabled* flag.

The *-cpu-idle* flag appends a column for each CPU specifying how much each CPU has been in an idle state during the sample. Similar to *-cpu-idle*, *-cpu-exec-mode* adds information about other CPU execution modes.

The *-cpu-host-ticks* flag appends a column for each CPU specifying how much each CPU taken from the host processor when being simulated. This feature does not currently work on Windows and is therefore disabled.

To see how much of the memory-limit that has been used the *-mem* can be used. Notice that this is only the “image” memory used, thus RAM, disk etc. So it cannot be used to find memory leaks for normal heap allocations. If this number goes down compared to the previous sample it means that memory-limit has been reached and Simics has swapped out dirty pages to disk.

Simics can share identical pages across multiple simulated targets. If the targets for instance run the same OS, Simics can keep one copy of a page instead of multiple copies, which consequently reduces host memory consumption. To see how much memory is saved, the *-shared* can be used. Notice that this figure only shows how much “image” memory that is saved. The page sharing mechanism can also reduce internal state, but this memory reduction is not accounted for.

The output contains a number of columns this is what they represent; *Total-vt* represent how many seconds (virtual time) that has been executed on the system in total. *Sample-vt* is the measurement sample time in seconds (virtual time). *Sample-rt* is how long the corresponding real time it took on Simics to execute. *Slowdown* represent the ratio between how fast Simics executes compared the simulated system. *CPU* indicates how much of the host CPU that was used for the sample, the figure should be close to 100%, otherwise another process is running at the same time or Simics stalls on paging.

Idle represent how much all CPUs in the system was in idle during the sample. A large idle percentage means that Simics can fast-forward time more, and hence gives better performance. *Disabled* shows how many CPUs and the percent of the total system which are currently not activated yet.

Provided By
[system-perfmeter](#)

[system-perfmeter-plot](#)

Synopsis

```
system-perfmeter-plot log_file html_file
```

Description

Takes a recorded system-perfmeter log *log_file*, and creates a HTML page *html_file* with relevant graphs. The graphs that are created plot overall load, CPU load, utilization, speed, effective MIPS, and memory usage as a function of both virtual time and real time. Corresponding virtual times are added as vertical lines in real time graphs, and the other way around for virtual time graphs.

Load as plotted by this command is defined as the non-idle portion of the execution, meaning that a 5 processor system where each processor is 75% idle would have an overall load of $5 * 25\%$, or 125%.

The memory usage graph shows the memory usage in % of the set memory limit. Note that the default memory-limit depends of the memory capacity of the host system, and that the memory-limit can also be changed dynamically.

The **system-perfmeter-plot** command requires the pychart package to be installed in the Simics Python environment. To install pychart, first download it from <http://download.gna.org/pychart/>. Then unpack it, and run `path-to-simics/bin/mini-python setup.py build` followed by `path-to-simics/bin/mini-python setup.py install` in your workspace. For the installation to work, you need to have write permission to the installation of the Simics Base package.

Provided By

[system-perfmeter](#)

tcpdump**Synopsis**

```
tcpdump [link] [probe] [device] [“flags”]  
tcpdump-stop [link] [probe]
```

Description

Runs the **tcpdump** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to **tcpdump**.

Provided By

[eth-links](#)

tcpdump-stop**Synopsis**

```
tcpdump-stop [link] [probe]  
tcpdump [link] [probe] [device] [“flags”]
```

Description

Runs the **tcpdump** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to **tcpdump**.

Provided By
[eth-links](#)

telnet-frontend

Synopsis

telnet-frontend [*port*] [*max-connections*] [-non-interactive]

Description

Creates a new Simics command-line accessible using telnet. If no TCP port is specified, a currently free port will be selected where port 8192 is tried first. If a busy port is specified the command will fail. The port actually used is returned by the command.

The `-non-interactive` flag can be used to prepare for scripted, non-interactive sessions. This prevents Simics from using fancy formatting and coloring of the output and inhibits redrawing of the prompt.

It is possible to limit the number of allowed connections to the frontend with the *max-connections* argument. Once this number of connections has been reached, no new connections are allowed even if previous ones disconnect. It is possible to reset the count of connections by writing directly to the *num_connections* attribute in the **telnet-frontend** class.

Provided By
[telnet-frontend](#)

trace-breakpoint

Synopsis

trace-breakpoint (*breakpoint|all|-list*)
untrace-breakpoint (*breakpoint|all|-list*)

Description

Enables and disables tracing of breakpoints. When enabled, breakpoint hits will be traced to the console instead of stopping the simulation.

The *id* parameter specifies the breakpoint to trace.

Instead of an id, the `-all` flag may be given. This will enable or disable tracing of all breakpoints.

Using the `-list` argument will print out the breakpoints currently being traced.

Provided By
[Simics Core](#)

See Also
[log-setup](#)

trace-cr

Synopsis

```
trace-cr ("register"|-all|-list)
<int_register>.trace-cr ("register"|-all|-list)
<int_register>.untrace-cr ("register"|-all|-list)
untrace-cr ("register"|-all|-list)
```

Description

Enables and disables tracing of control register updates. When this is enabled, every time the specified control register is updated during simulation a message is printed. The message will name the register being updated, and the new value. The new value will be printed even if it is identical to the previous value.

The *reg-name* parameter specifies which control register should be traced. The available control registers depends on the simulated target.

Instead of a register name, the `-all` flag may be given. This will enable or disable tracing of all control register.

Using the `-list` argument will print out the registers accesses currently being traced. The non-namespace variant of this command traces control register updates on all processors of the same class in the same group as the currently inspected processor. The namespace variant affects only the selected processor.

Provided By

[Simics Core](#)

See Also

[break-cr](#), [log-setup](#)

trace-exception**Synopsis**

```
trace-exception (number|"name"|-all|-list)
```

Description

Enables and disables tracing of exceptions. When this is enabled, every time the specified exception occurs during simulation a message is printed.

The *exception* parameter specifies which exception should be traced. The available exceptions depends on the simulated target.

Instead of an exception, the `-all` flag may be given. This will enable or disable tracing of all exceptions.

Using the `-list` argument will print out the exceptions currently being traced.

The non-namespace variant of this command traces exceptions on all processors of the same class in the same group as the currently inspected processor. The namespace variant affects only the selected processor.

Provided By

[Simics Core](#)

See Also

[break-exception](#), [log-setup](#)

trace-hap**Synopsis**

trace-hap ("hap"|-all|-list)
untrace-hap ("hap"|-all|-list)

Description

Enables and disables tracing of haps. When this is enabled, every time the specified hap is triggered a message is printed.

The *hap* parameter specifies the hap.

Instead of a hap, the `-all` flag may be given. This will enable or disable tracing of all haps.

Using the `-list` argument will print out the haps currently being traced.

Provided By

[Simics Core](#)

See Also

[break-hap](#), [list-haps](#), [log-setup](#)

trace-io**Synopsis**

trace-io ("device"|-all|-list)
untrace-io ("device"|-all|-list)

Description

Enables and disables tracing of device accesses. When this is enabled, every time the specified device is accessed during simulation a message is printed. The data value is interpreted using the default endian of the processor initiating the access or the current processor for device initiated operations. The endian used is reported in the output as BE (big-endian) or LE (little-endian).

The *device* parameter specifies the device object that should be traced.

Instead of an object name, the `-all` flag may be given. This will enable or disable tracing of all devices.

Using the `-list` argument will print out the objects currently being traced.

Provided By

[Simics Core](#)

See Also

[break-io](#), [log-setup](#), [trace-io-setup](#)

trace-io-setup**Synopsis**

```
trace-io-setup [-default-little-endian] [-default-big-endian]
```

Description

The `-default-little-endian` and `-default-big-endian` flags specify the byte order to use when none is implied by the context.

If no argument is given, the current setting is shown.

Provided By

[Simics Core](#)

See Also

[trace-io](#)

try**Synopsis**

```
try { commands } except { on-error-commands }  

get-error-command  

get-error-file  

get-error-line  

get-error-message
```

Description

Runs all *commands* from the **try** part until an error occurs. If no error is encountered, *on-error-commands* are ignored, but if an error does occur the *on-error-commands* are run. Information about the error is available in the **except** part through the **get-error-commands**, **get-error-message**, **get-error-file** and **get-error-line** commands.

Provided By

[Simics Core](#)

turbo-disable-block-profile**Synopsis**

```
turbo-disable-block-profile  

turbo-enable-block-profile
```

Description

Enable or disable turbo self profiling.

Provided By

[symtable](#)

turbo-enable-block-profile

Synopsis

turbo-enable-block-profile
turbo-disable-block-profile

Description

Enable or disable turbo self profiling.

Provided By

[symtable](#)

See also

[turbo-show-block-profile](#)

turbo-info**Synopsis**

turbo-info

Description

Print turbo statistics for all classes and CPUs.

Provided By

[symtable](#)

turbo-info-clean**Synopsis**

turbo-info-clean

Description

Reset turbo statistics. Also resets exec-info stats.

Provided By

[symtable](#)

turbo-show-block-profile**Synopsis**

turbo-show-block-profile [*num*] [*filename*] [-sort-target-address] [-target-disassemble]
[-show-host-address] [-sort-host-address]

Description

List the *num* hottest turbo blocks. The default when *num* is not given is to print all blocks that have been hit by a sample.

Provided By

[symtable](#)

See also

[turbo-enable-block-profile](#)

unbreak**Synopsis**

unbreak (*id|all*) *address length [-r] [-w] [-x]*

Description

Removes an address range from a breakpoint, splitting the breakpoint if necessary.

-r (read), *-w* (write) and *-x* (execute) specify the type of breakpoint that should be removed in the given address range. Default is *execute*.

id is the ID number of the breakpoint to remove. The *-all* flag removes all breakpoints of the specified access type within the specified address range. Default access type is *execute*.

Use **list-breakpoints** to list the IDs of breakpoints.

Provided By

[Simics Core](#)

See Also

[`<breakpoint>.break, delete`](#)

unbreak-cr**Synopsis**

unbreak-cr ("register"|-all|-list)
<int_register>.break-cr ("register"|-all|-list)
<int_register>.unbreak-cr ("register"|-all|-list)
break-cr ("register"|-all|-list)

Description

Enables and disables breaking simulation on control register updates. When this is enabled, every time the specified control register is updated during simulation a message is printed. The message will name the register being updated, and the new value. The new value will be printed even if it is identical to the previous value.

The *reg-name* parameter specifies which control register should be traced. The available control registers depends on the simulated target.

Instead of a register name, the *-all* flag may be given. This will enable or disable tracing of all control register.

Using the *-list* argument will print out the registers accesses on which a breakpoint will trigger.

The non-namespace variant of this command breaks the simulation on control register updates on all processors of the same class in the same group as the currently inspected processor. The namespace variant affects only the selected processor.

Provided By[Simics Core](#)**See Also**[trace-cr](#), [`<breakpoint>.break`](#), [log-setup](#)**unbreak-exception****Synopsis****unbreak-exception** (*number*|“*name*”|-all|-list)**Description**

Enables and disables breaking simulation on exceptions. When this is enabled, every time the specified exception occurs during simulation a message is printed.

The *exception* parameter specifies which exception should be traced. The available exceptions depends on the simulated target.

Instead of an exception, the `-all` flag may be given. This will enable or disable tracing of all exceptions.

Using the `-list` argument will print out the exceptions on which a breakpoint will trigger.

The non-namespace variant of this command installs breakpoints on all processors of the same class in the same group as the currently inspected processor. The namespace variant affects only the selected processor.

Provided By[Simics Core](#)**See Also**[trace-exception](#), [`<breakpoint>.break`](#)**unbreak-hap****Synopsis****unbreak-hap** (“*hap*”|-all|-list)**break-hap** (“*hap*”|-all|-list)**Description**

Enables and disables breaking simulation on haps. When this is enabled, every time the specified hap is triggered a message is printed and simulation is stopped.

The *hap* parameter specifies the hap.

Instead of a hap, the `-all` flag may be given. This will enable or disable breaking on all haps.

Using the `-list` argument will print out the haps on which a breakpoint will trigger.

Provided By
[Simics Core](#)

See Also
[trace-hap](#), [list-haps](#)

unbreak-io

Synopsis
unbreak-io ("device"-all|-list)
break-io ("device"-all|-list)

Description

Enables and disables breaking simulation on device accesses. When this is enabled, every time the specified device is accessed during simulation a message is printed and the simulation stops.

The *device* parameter specifies which device object should be traced.

Instead of an object name, the `-all` flag may be given. This will enable or disable breaking on accesses to all device.

Using the `-list` argument will print out the objects for which I/O breakpoints will trigger.

Provided By
[Simics Core](#)

See Also
[trace-io](#), <breakpoint>.break

uncall-function

Alias
 uncall

Synopsis
uncall-function

Description

uncall-function causes the simulation to run backwards until just before the current function was called.

Provided By
[Simics Core](#)

See Also
[finish-function](#), [next-instruction](#), [next-line](#), [reverse-next-instruction](#), [reverse-next-line](#),
[reverse-step-instruction](#), [reverse-step-line](#), [step-instruction](#), [step-line](#)

undisplay**Synopsis****undisplay** *expression-id***Description**

Remove a Python expression, or a frontend statement that was previously installed with the `display` command. The argument is the id number of the expression, as listed by `display -l`.

Provided By[Simics Core](#)**See Also**[display](#)**unload-module****Synopsis****unload-module** "*module*"**Description**

Deprecated command that will be removed in the next major version of Simics.

Provided By[Simics Core](#)**unset****Synopsis****unset** [-a] ["*variables*" ...]**Description**

Removes (unsets) a CLI variable. The `-a` flag causes all variables to be removed, *except* the ones specified as *variables*.

Provided By[Simics Core](#)**untrace-breakpoint****Synopsis**

untrace-breakpoint (*breakpoint*|`-all`|`-list`)
trace-breakpoint (*breakpoint*|`-all`|`-list`)

Description

Enables and disables tracing of breakpoints. When enabled, breakpoint hits will be traced to the console instead of stopping the simulation.

The *id* parameter specifies the breakpoint to trace.

Instead of an id, the `-all` flag may be given. This will enable or disable tracing of all breakpoints.

Using the `-list` argument will print out the breakpoints currently being traced.

Provided By

[Simics Core](#)

See Also

[log-setup](#)

untrace-cr**Synopsis**

```
untrace-cr ("register"|-all|-list)
<int_register>.trace-cr ("register"|-all|-list)
<int_register>.untrace-cr ("register"|-all|-list)
trace-cr ("register"|-all|-list)
```

Description

Enables and disables tracing of control register updates. When this is enabled, every time the specified control register is updated during simulation a message is printed. The message will name the register being updated, and the new value. The new value will be printed even if it is identical to the previous value.

The *reg-name* parameter specifies which control register should be traced. The available control registers depends on the simulated target.

Instead of a register name, the `-all` flag may be given. This will enable or disable tracing of all control register.

Using the `-list` argument will print out the registers accesses currently being traced.

The non-namespace variant of this command traces control register updates on all processors of the same class in the same group as the currently inspected processor. The namespace variant affects only the selected processor.

Provided By

[Simics Core](#)

See Also

[break-cr](#), [log-setup](#)

untrace-exception

Synopsis

untrace-exception (*number*|“*name*”|-all|-list)

Description

Enables and disables tracing of exceptions. When this is enabled, every time the specified exception occurs during simulation a message is printed.

The *exception* parameter specifies which exception should be traced. The available exceptions depends on the simulated target.

Instead of an exception, the `-all` flag may be given. This will enable or disable tracing of all exceptions.

Using the `-list` argument will print out the exceptions currently being traced.

The non-namespace variant of this command traces exceptions on all processors of the same class in the same group as the currently inspected processor. The namespace variant affects only the selected processor.

Provided By

[Simics Core](#)

See Also

[break-exception](#), [log-setup](#)

untrace-hap**Synopsis**

untrace-hap (“*hap*”|-all|-list)
trace-hap (“*hap*”|-all|-list)

Description

Enables and disables tracing of haps. When this is enabled, every time the specified hap is triggered a message is printed.

The *hap* parameter specifies the hap.

Instead of a hap, the `-all` flag may be given. This will enable or disable tracing of all haps.

Using the `-list` argument will print out the haps currently being traced.

Provided By

[Simics Core](#)

See Also

[break-hap](#), [list-haps](#), [log-setup](#)

untrace-io**Synopsis**

untrace-io (“*device*”|-all|-list)
trace-io (“*device*”|-all|-list)

Description

Enables and disables tracing of device accesses. When this is enabled, every time the specified device is accessed during simulation a message is printed. The data value is interpreted using the default endian of the processor initiating the access or the current processor for device initiated operations. The endian used is reported in the output as BE (big-endian) or LE (little-endian).

The *device* parameter specifies the device object that should be traced.

Instead of an object name, the `-all` flag may be given. This will enable or disable tracing of all devices.

Using the `-list` argument will print out the objects currently being traced.

Provided By

[Simics Core](#)

See Also

[break-io](#), [log-setup](#), [trace-io-setup](#)

up

Synopsis

up [*N*]

Description

Moves *N* frames up the stack (towards the outermost frame). *N* defaults to one.

Provided By

[Simics Core](#)

See Also

[frame](#), [down](#), [stack-trace](#)

version

Synopsis

version [-v]

Description

Prints the Simics version. With the `-v` flag, some additional information is printed.

Provided By

[Simics Core](#)

wait-for-breakpoint

Synopsis

wait-for-breakpoint [-always] *breakpoint-id*

Description

Postpones execution of a script branch until a memory breakpoint with the supplied id is triggered. The name of the object initiating the memory access that caused the breakpoint to trigger is returned. The *-always* flag is internal and should not be used.

Provided By

[Simics Core](#)

See Also

[script-branch](#), [break](#)

wait-for-hap — deprecated**Synopsis**

wait-for-hap "hap" [*object*] [*idx0*] [*idx1*] ["*ret*"]

Description

This command is deprecated; use [script-branch](#), [wait-for-breakpoint](#), [<text-console>.wait-for-string](#), [<int_register>.wait-for-register-read](#), [<int_register>.wait-for-register-write](#), [<cycle>.wait-for-cycle](#), [<step>.wait-for-step](#) or [<os-awareness>.wait-for-enter](#) instead.

Provided By

[Simics Core](#)

wait-for-script-barrier**Synopsis**

wait-for-script-barrier *barrier*

Description

Suspends execution of a script branch until enough script branches have entered the script barrier *barrier*.

Provided By

[Simics Core](#)

See Also

[script-branch](#), [create-script-barrier](#)

wait-for-script-pipe**Synopsis**

wait-for-script-pipe *pipe*

Description

Suspends execution of a script branch until there is some data to read from a script pipe. If there already is data available, the command will return immediately. The return value is the data send by the [add-data-to-script-pipe](#) commands.

Provided By
[Simics Core](#)

See Also

[script-branch](#), [create-script-pipe](#), [script-pipe-has-data](#), [add-data-to-script-pipe](#)

wait-for-variable — deprecated

Synopsis
wait-for-variable “*variable*” [*value*]

Description

This command is deprecated; use [wait-for-script-pipe](#) or [wait-for-script-barrier](#) instead.

Provided By
[Simics Core](#)

whereis

Synopsis
whereis [-d] *address*

Description

Displays the function or variable and (if possible) the source file and line number corresponding to *address*. If –d is specified, searches only data symbols. Does not look for stack-allocated variables.

Provided By
[Simics Core](#)

See Also
[pos](#), [psym](#)

while

Synopsis
while *condition* { *commands* }

Description

Runs a block of commands while *condition* is true.

Provided By
[Simics Core](#)

See Also
[if](#), [foreach](#)

win-about**Synopsis****win-about****Description**

Shows information about why the GUI libraries such as wxPython failed to load. This command will only replace the real **win-about** on a failure.

Provided By[Simics Core](#)**wireshark****Alias**

ethereal

Synopsis

```
wireshark [link] [probe] [device] [“flags”]
wireshark-stop [link] [probe]
```

Description

Runs the **wireshark** or **ethereal** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to program.

Provided By[eth-links](#)**wireshark-stop****Alias**

ethereal-stop

Synopsis

```
wireshark-stop [link] [probe]
wireshark [link] [probe] [device] [“flags”]
```

Description

Runs the **wireshark** or **ethereal** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to program.

Provided By[eth-links](#)**write-configuration**

Synopsis

```
write-configuration file [-z] [-u] [“comment”]
```

Description

Write the current configuration to *file*. Note that some classes save their state incrementally; files from earlier checkpoints (or from the starting configuration) may be referenced in the new checkpoint.

Use the **-z** flag for compressed images, or **-u** for uncompressed. The default is taken from the preference object.

To add a description to the checkpoint, use the *comment* argument. The comment is saved in the `info` file in the checkpoint bundle.

Provided By

[Simics Core](#)

See Also

[read-configuration](#), [save-persistent-state](#)

write-reg**Synopsis**

```
write-reg [cpu-name] “reg-name” value  

<int_register>.write-reg “reg-name” value
```

Description

Use this command to set the value of a register on an object implementing the `int_register` interface. For example, to set the `eax` register on the x86 processor `cpu0` to 3, write **write-reg** `cpu0 eax 3`. You can also use the method variant: `cpu0.write-reg eax 3`.

This function may or may not have the correct side-effects, depending on target and register. If no *cpu-name* is given, the current frontend processor is used.

Provided By

[Simics Core](#)

See Also

`%`, [read-reg](#), [pregs](#), [pselect](#)

x**Synopsis**

```
x [cpu-name] address [size] [-c]  

<memory-space>.x address [size] [-c]
```

Description

Display the contents of a memory space starting at *address*. Either the memory space is explicitly specified as in `<memory-space>.x` or the CPU connected to the memory space can be specified; e.g., `<processor>.x`. By itself, `x` operates on the memory connected to the current frontend processor.

If the memory is accessed via a CPU, the type of *address* is specified by a prefix. For physical addresses use `p:address`; for virtual addresses, `v:address` on non-x86 targets. On x86, use `segment-register:offset` or `l:address` for x86 linear addresses.

If no prefix is given it will be interpreted as a virtual address. On x86 the default is `ds:address` (data segment addressing).

The access will be made in inquiry mode, which means it will have no side-effects on the CPU or the accessed object. Use the `<memory-space>.read` command to do non-inquiry accesses.

The *size* argument specifies the number of bytes to examine. When examining virtual memory, only addresses which can be found in the TLB or hardware page tables (if any) are shown. Unmapped addresses are shown as “--”, undefined physical addresses as “***”.

The `-c` flag compresses the output by not displaying sequences of zeros.

Provided By

[Simics Core](#)

See Also

[disassemble](#), [get](#), [set](#)

|

Synopsis

`arg1 | arg2`

Description

Bitwise OR operation

Provided By

[Simics Core](#)

~

Synopsis

`~ arg1`

Description

Bitwise not.

Provided By

[Simics Core](#)

Chapter 4

Components

cell-and-clocks

Provided By

[clock](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “cell_and_clock” component builds a simulation cell with a configurable number of clocks. Each clock is exported as a connector. This component is meant to be used for building small test configurations

Connectors

Name	Type	Direction
clock0	clock	down

Attributes

clock_number

Optional attribute; **read/write** access; type: `integer`. Number of clocks to run

components

Optional attribute; **read/write** access; type: `[o*]`. List of components below the top-level component. This attribute is not valid until the object has been instantiated.

cpu_list

Pseudo attribute; **read/write** access; type: `[o*]`. List of all processors below the top-level component. This attribute is not valid until the object has been instantiated.

freq_mhz

Optional attribute; **read/write** access; type: `integer`. Clocks’ frequency

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

machine_icon

Optional attribute; **read/write** access; type: `string` or `nil`. An instance of a top-level component may override the default *system_icon* with its own icon. This attribute is the name of an 80x80 pixel large icon in PNG format that should reside in the `[host]/lib/images` directory of the Simics installation or the workspace.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

system_info

Optional attribute; **read/write** access; type: `string` or `nil`. A short single-line description of the current configuration of the system that the component is a top-level of. The line may include the Unix name of the simulated machine, the installed operating system, or similar information. For example “Tango - Fedora Core 5 Linux”.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

system_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of an 80x80 pixels large icon in PNG format used to graphically represent the system that the component is a top-level of.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

`<cell-and-clocks>.info`

Synopsis

`<cell-and-clocks>.info`

Description

Print detailed information about the configuration of the device.

`<cell-and-clocks>.status`

Synopsis

<cell-and-clocks>.status

Description

Print detailed information about the current status of the device.

component

Provided By
[Simics Core](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

Base component class, should not be instantiated.

Attributes

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

<code>connect</code>	<i>deprecated</i> — connect components
<code>debug-program</code>	<i>deprecated</i> — debug program
<code>delete</code>	delete non-instantiated components
<code>disconnect</code>	disconnect component connector
<code>get-component-object</code>	get named object from components
<code>get-connection</code>	return connection information
<code>get-connector-list</code>	return list of connectors
<code>get-processor-list</code>	return list of processors
<code>info</code>	print information about the device
<code>status</code>	print status of the device

Command Descriptions

`<component>.connect — deprecated`

Synopsis

`<component>.connect ["connector"] component ["dst-connector"] [-f]`

Description

This command is deprecated; use '`connect <cnt0> <cnt1>`' instead.

Connects the *connector* of this component to the *dst-connector* connector of *component*. Indexed connectors must be quoted. The *f* flag selects the first unused connector (in alphabetic order).

`<component>.debug-program — deprecated`

Synopsis

`<component>.debug-program file [pid]`

Description

This command is deprecated; use `<os_awareness>.track` instead.

Debug a program running on the component. The command takes a single argument: the program file on the host. This file must have the same base name as the program on the target. The command will set up Simics for symbolic debugging of the next process which executes the program.

If you supply the pid argument to the command the command will set up symbolic debugging for the process on the target with that pid, instead of the next process which executes the program.

`<component>.delete`

Synopsis

`<component>.delete`

Description

Deletes the component. The component may not be connected to any other component. Deleting a non-instantiated component should work without any problem. Deleting an instantiated component requires that all objects in the component supports deletion.

`<component>.disconnect`

Synopsis

```
<component>.disconnect ["connector"] [component] ["dst-connector"] [-quiet]
```

Description

Disconnects the *connector* from another component connector. Connectors can only be disconnected if they support hotplugging.

`<component>.get-component-object`

Synopsis

```
<component>.get-component-object "object"
```

Description

Get the configuration object with name *object* from the component. This is similar to the `.` operator.

`<component>.get-connection`

Synopsis

```
<component>.get-connection "connector"
```

Description

Return information about the component connected to the selected *connector*. The return value is either an empty list, a list of two items, or a list of a list of two items for multi connections. The list of two items are the name of the other component and the other connector.

`<component>.get-connector-list`

Synopsis

```
<component>.get-connector-list ["connector-type"] [-c] [-u]
```

Description

Return a list of all connectors for the component. If *connector-type* is given, only connectors of that type are returned. The `-c` flag limits the list to connected connectors and `-u` only return unconnected ones.

`<component>.get-processor-list`

Synopsis

```
<component>.get-processor-list
```

Description

Return a list of all processors that are part of the component hierarchy defined by this top-level component. This command is only applicable to top-level components.

```
<component>.info
```

Synopsis

```
<component>.info
```

Description

Print detailed information about the configuration of the device.

```
<component>.status
```

Synopsis

```
<component>.status
```

Description

Print detailed information about the current status of the device.

cp3_quad100tx

Provided By

[cp3_quad100tx](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The cp3-quad100tx component class.

Connectors

Name	Type	Direction
pci	pci-bus	up
eth[1-4]	ethernet-link	down

Attributes

mac_addr1

Required attribute; **read/write** access; type: `string`. The MAC address of eth1

mac_addr2

Required attribute; **read/write** access; type: `string`. The MAC address of eth2

mac_addr3

Required attribute; **read/write** access; type: `string`. The MAC address of eth3

mac_addr4

Required attribute; **read/write** access; type: `string`. The MAC address of eth4

Command List

Commands

[**info**](#) print information about the device

[**status**](#) print status of the device

Command Descriptions

`<cp3_quad100tx>.info`

Synopsis

`<cp3_quad100tx>.info`

Description

Print detailed information about the configuration of the device.

<cp3_quad100tx>.status

Synopsis

<cp3_quad100tx>.status

Description

Print detailed information about the current status of the device.

datagram_link

Provided By
[datagram-link](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The datagram link component creates a datagram-link, which is a simple broadcast bus forwarding messages (as sequences of bytes) from a sender device to all other devices present on the link. The datagram-link is both an example of how to build a link with the Simics Link Library, and a simple broadcast link that can be reused when multi-cell communication between devices is necessary. Refer to the *Link Library Programming Guide* for more information.

Connectors

Name	Type	Direction
device0	datagram-link	any

Attributes

global_id

Optional attribute; **read/write** access; type: `string` or `nil`. Global identifier for use in distributed simulation or `NIL` if the link is not distributed.

goal_latency

Optional attribute; **read/write** access; type: `float`. Goal latency in seconds for this link. See also the `set-min-latency` command.

Command List

Commands

[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

<datagram_link>.info

Synopsis

<datagram_link>.info

Description

Print detailed information about the configuration of the device.

<datagram_link>.status

Synopsis

<datagram_link>.status

Description

Print detailed information about the current status of the device.

ddr-memory-module

Provided By
[memory-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “ddr-memory-module” component represents a DDR memory module.

Connectors

Name	Type	Direction
mem_bus	mem-bus	up

Attributes

banks

Optional attribute; **read/write** access; type: `integer`. Number of banks.

cas_latency

Optional attribute; **read/write** access; type: `integer`. CAS-latency; each set bit corresponds to a latency the memory can handle

columns

Optional attribute; **read/write** access; type: `integer`. Number of columns.

ecc_width

Optional attribute; **read/write** access; type: `integer`. The error correction width.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

module_data_width

Optional attribute; **read/write** access; type: `integer`. The module SDRAM width.

module_type

Optional attribute; **read/write** access; type: `string`. Type of memory.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

primary_width

Optional attribute; **read/write** access; type: `integer`. Primary SDRAM width.

rank_density

Optional attribute; **read/write** access; type: `integer`. The rank density.

ranks

Optional attribute; **read/write** access; type: `integer`. Number of ranks (logical banks).

rows

Optional attribute; **read/write** access; type: `integer`. Number of rows.

speed

Optional attribute; **read/write** access; type: `string`. PC standard speed. Supported values are PC2700 and none.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<ddr-memory-module>.info

Synopsis

<ddr-memory-module>.info

Description

Print detailed information about the configuration of the device.

<ddr-memory-module>.status

Synopsis

<ddr-memory-module>.status

Description

Print detailed information about the current status of the device.

ddr2-memory-module

Provided By
[memory-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “ddr2-memory-module” component represents a DDR2 memory module.

Connectors

Name	Type	Direction
mem_bus	mem-bus	up

Attributes

banks

Optional attribute; **read/write** access; type: `integer`. Number of banks.

cas_latency

Optional attribute; **read/write** access; type: `integer`. CAS-latency; each set bit corresponds to a latency the memory can handle

columns

Optional attribute; **read/write** access; type: `integer`. Number of columns.

ecc_width

Optional attribute; **read/write** access; type: `integer`. The error correction width.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

module_data_width

Optional attribute; **read/write** access; type: `integer`. The module SDRAM width.

module_type

Optional attribute; **read/write** access; type: `string`. Type of memory.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

primary_width

Optional attribute; **read/write** access; type: `integer`. Primary SDRAM width.

rank_density

Optional attribute; **read/write** access; type: `integer`. The rank density.

ranks

Optional attribute; **read/write** access; type: `integer`. Number of ranks (logical banks).

rows

Optional attribute; **read/write** access; type: `integer`. Number of rows.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

`<ddr2-memory-module>.info`

Synopsis

`<ddr2-memory-module>.info`

Description

Print detailed information about the configuration of the device.

`<ddr2-memory-module>.status`

Synopsis

<ddr2-memory-module>.status

Description

Print detailed information about the current status of the device.

ddr3-memory-module

Provided By
[memory-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “ddr3-memory-module” component represents a DDR3 memory module.

Connectors

Name	Type	Direction
mem_bus	mem-bus	up

Attributes

banks

Optional attribute; **read/write** access; type: `integer`. Number of banks.

cas_latency

Optional attribute; **read/write** access; type: `integer`. CAS-latency; each set bit corresponds to a latency the memory can handle

columns

Optional attribute; **read/write** access; type: `integer`. Number of columns.

ecc_width

Optional attribute; **read/write** access; type: `integer`. The error correction width.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

module_data_width

Optional attribute; **read/write** access; type: `integer`. The module SDRAM width.

module_type

Optional attribute; **read/write** access; type: `string`. Type of memory.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

primary_width

Optional attribute; **read/write** access; type: `integer`. Primary SDRAM width.

rank_density

Optional attribute; **read/write** access; type: **integer**. The rank density.

ranks

Optional attribute; **read/write** access; type: **integer**. Number of ranks (logical banks).

rows

Optional attribute; **read/write** access; type: **integer**. Number of rows.

top_component

Optional attribute; **read/write** access; type: **object or nil**. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: **string**. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: **string or nil**. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: **boolean or nil**. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<ddr3-memory-module>.info

Synopsis

<ddr3-memory-module>.info

Description

Print detailed information about the configuration of the device.

<ddr3-memory-module>.status

Synopsis

<ddr3-memory-module>.status

Description

Print detailed information about the current status of the device.

dummy-component

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

Dummy component used for configurations that are not component based.

Attributes

components

Optional attribute; **read/write** access; type: [o*]. List of components below the top-level component. This attribute is not valid until the object has been instantiated.

cpu_list

Pseudo attribute; **read/write** access; type: [o*]. List of all processors below the top-level component. This attribute is not valid until the object has been instantiated.

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

machine_icon

Optional attribute; **read/write** access; type: string or nil. An instance of a top-level component may override the default *system_icon* with its own icon. This attribute is the name of an 80x80 pixel large icon in PNG format that should reside in the [host]/lib/images directory of the Simics installation or the workspace.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

system_info

Optional attribute; **read/write** access; type: string or nil. A short single-line description of the current configuration of the system that the component is a top-level of. The line may include the Unix name of the simulated machine, the installed operating system, or similar information. For example “Tango - Fedora Core 5 Linux”.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

system_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of an 80x80 pixels large icon in PNG format used to graphically represent the system that the component is a top-level of.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

`<dummy-component>.info`

Synopsis

`<dummy-component>.info`

Description

Print detailed information about the configuration of the device.

`<dummy-component>.status`

Synopsis

`<dummy-component>.status`

Description

Print detailed information about the current status of the device.

etg_comp

Provided By
[std-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The etg component represents an Ethernet traffic generator.

Connectors

Name	Type	Direction
connector_ethernet	ethernet-link	down

Attributes

dst_ip

Required attribute; **read/write** access; type: `string`. Destination IP address for generated traffic.

ip

Required attribute; **read/write** access; type: `string`. IP address of the traffic generator.

netmask

Required attribute; **read/write** access; type: `string`. IP netmask of the traffic generator.

Command List

Commands

[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

`<etg_comp>.info`

Synopsis

`<etg_comp>.info`

Description

Print detailed information about the configuration of the device.

`<etg_comp>.status`

Synopsis

<etg_comp>.status

Description

Print detailed information about the current status of the device.

etg_panel

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [conf_object](#), [log_object](#)

Description
The Ethernet Generator System Panel.

Command List

Commands
[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

<etg_panel>.info

Synopsis
[**<etg_panel>.info**](#)

Description
Print detailed information about the configuration of the device.

<etg_panel>.status

Synopsis
[**<etg_panel>.status**](#)

Description
Print detailed information about the current status of the device.

eth_injector_comp

Provided By

[eth_injector_comp](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The Ethernet frame injector is a pseudo-device that reads a pcap formatted file and inject the packets it found into another device, or an Ethernet link.

Connectors

Name	Type	Direction
link	ethernet-link	up

Command List

Commands

[info](#) print information about the device

[status](#) print status of the device

Command Descriptions

<eth_injector_comp>.info

Synopsis

[`<eth_injector_comp>.info`](#)

Description

Print detailed information about the configuration of the device.

<eth_injector_comp>.status

Synopsis

[`<eth_injector_comp>.status`](#)

Description

Print detailed information about the current status of the device.

ethernet_cable

Provided By
[eth-links](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

Ethernet cable: this component represents a two-points ethernet cable, allowing two devices to connect to each other.

Connectors

Name	Type	Direction
device0	ethernet-link	any

Attributes

global_id

Optional attribute; **read/write** access; type: `string` or `nil`. Global identifier for use in distributed simulation or `NIL` if the link is not distributed.

goal_latency

Optional attribute; **read/write** access; type: `float`. Goal latency in seconds for this link. See also the `set-min-latency` command.

Command List

Commands

connect-real-network-bridge	connect to the real network
connect-real-network-host	connect to the real network
connect-real-network-napt	enable NAPT from simulated network
connect-real-network-router	<i>deprecated</i> — connect to the real network
disconnect-real-network	disconnect from the real network
get-free-connector	return the name of an unused connector
info	print information about the device
pcap-dump	Dump network traffic to a file, in libpcap format
pcap-dump-stop	stop the current dump
set-goal-latency	Set the link's goal latency (in seconds).
status	print status of the device
tcpdump	run the tcpdump program
tcpdump-stop	stop the current tcpdump capture
wireshark	run the wireshark/ethereal program
wireshark-stop	stop the current wireshark capture

Command Descriptions

<ethernet_cable>.connect-real-network-bridge

Synopsis

```
<ethernet_cable>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
<ethernet-link>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
<ethernet_hub>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
<ethernet_switch>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
```

Description

Creates an Ethernet bridge between a simulated Ethernet link and a real network through an Ethernet interface of the simulation host.

The optional *interface* argument specifies the Ethernet or TAP interface of the host to use.

By default a TAP interface is used, but if the *host-access* argument is `raw`, raw access to an Ethernet interface is used.

MAC address translation can be disabled with the `-no-mac-xlate` flag.

The `-persistent` is for backward compatibility and should not be used.

If `-propagate-link-status` is specified, link status changes on the host interface will be propagated to all devices on the link that implements the link-status interface. For TAP, only 'up' and 'down' status will be propagated (and not 'unconnected'). Link status propagation is only supported on Linux.

This command returns the new real-network component object.

See the *Connecting to a Real Network* chapter of the *Ethernet Networks in Simics* manual for more information about how to connect to a real network.

See Also

[connect-real-network](#), [connect-real-network-host](#), [disconnect-real-network](#)

<ethernet_cable>.connect-real-network-host

Synopsis

```
<ethernet_cable>.connect-real-network-host ["interface"] [-persistent]
<ethernet-link>.connect-real-network-host ["interface"] [-persistent]
```

```
<ethernet_hub>.connect-real-network-host ["interface"] [-persistent]
<ethernet_switch>.connect-real-network-host ["interface"] [-persistent]
connect-real-network-host ["interface"] [-persistent]
```

Description

Connects a TAP interface of the simulation host to a simulated Ethernet link.

The optional *interface* argument specifies the TAP interface of the host to use.

The *-persistent* is for backward compatibility and should not be used.

This command returns the new real-network component object.

See the *Connecting to a Real Network* chapter of the *Simics User Guide* for more information about how to connect to a real network.

See Also

[connect-real-network](#), [connect-real-network-bridge](#), [disconnect-real-network](#)

```
<ethernet_cable>.connect-real-network-napt
```

Synopsis

```
<ethernet_cable>.connect-real-network-napt [service-node]
<ethernet_link>.connect-real-network-napt [service-node]
<ethernet_hub>.connect-real-network-napt [service-node]
<ethernet_switch>.connect-real-network-napt [service-node]
<ethernet_vlan_switch>.connect-real-network-napt [service-node]
connect-real-network-napt ethernet-link [service-node]
```

Description

Enables machines on the simulated network to initiate accesses to real hosts without the need to configure the simulated machine with a real IP address. NAPT (Network Address Port Translation) uses the IP address and a port number of the host that Simics is running on to perform the access. Replies are then translated back to match the request from the simulated machine. This command also enables NAPT for accesses that are initiated from the simulated machine.

See Also

[connect-real-network](#), [connect-real-network-port-in](#), [connect-real-network-port-out](#), [connect-real-network-host](#), [connect-real-network-bridge](#)

```
<ethernet_cable>.connect-real-network-router — deprecated
```

Synopsis

```
<ethernet_cable>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]
<ethernet_link>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]
<ethernet_hub>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]
```

```
<ethernet_switch>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]
connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]
```

Description

This command is deprecated; use [connect-real-network-bridge](#) or [connect-real-network-host](#) instead.

Deprecated command

See Also

[connect-real-network](#), [connect-real-network-host](#), [connect-real-network-bridge](#), [disconnect-real-network](#)

```
<ethernet_cable>.disconnect-real-network
```

Synopsis

```
<ethernet_cable>.disconnect-real-network
<ethernet-link>.disconnect-real-network
<ethernet_hub>.disconnect-real-network
<ethernet_switch>.disconnect-real-network
disconnect-real-network
```

Description

Closes all connections to real networks except port forwarding and NAPT.

See Also

[connect-real-network-host](#), [connect-real-network-bridge](#)

```
<ethernet_cable>.get-free-connector
```

Synopsis

```
<ethernet_cable>.get-free-connector
```

Description

This command returns the name of a connector which is not connected to anything.

```
<ethernet_cable>.info
```

Synopsis

```
<ethernet_cable>.info
```

Description

Print detailed information about the configuration of the device.

```
<ethernet_cable>.pcap-dump
```

Synopsis

```
<ethernet_cable>.pcap-dump file
<ethernet_cable>.pcap-dump-stop
```

Description

```
<ethernet_cable>.pcap-dump-stop
```

Synopsis

```
<ethernet_cable>.pcap-dump-stop
<ethernet_cable>.pcap-dump file
```

Description

```
<ethernet_cable>.set-goal-latency
```

Synopsis

```
<ethernet_cable>.set-goal-latency latency
```

Description

```
<ethernet_cable>.status
```

Synopsis

```
<ethernet_cable>.status
```

Description

Print detailed information about the current status of the device.

```
<ethernet_cable>.tcpdump
```

Synopsis

```
<ethernet_cable>.tcpdump [“flags”]
<ethernet_cable>.tcpdump-stop
```

Description

Runs the **tcpdump** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to **tcpdump**.

```
<ethernet_cable>.tcpdump-stop
```

Synopsis

```
<ethernet_cable>.tcpdump-stop  
<ethernet_cable>.tcpdump ["flags"]
```

Description

Runs the **tcpdump** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to **tcpdump**.

<ethernet_cable>.wireshark

Alias

```
<ethernet_cable>.ethereal
```

Synopsis

```
<ethernet_cable>.wireshark ["flags"]  
<ethernet_cable>.wireshark-stop
```

Description

Runs the **wireshark** or **ethereal** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to program.

<ethernet_cable>.wireshark-stop

Alias

```
<ethernet_cable>.ethereal-stop
```

Synopsis

```
<ethernet_cable>.wireshark-stop  
<ethernet_cable>.wireshark ["flags"]
```

Description

Runs the **wireshark** or **ethereal** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to program.

ethernet_hub

Provided By
[eth-links](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

Ethernet hub: this component represents a simple broadcasting ethernet link allowing any number of devices to connect.

Connectors

Name	Type	Direction
device0	ethernet-link	any

Attributes

global_id

Optional attribute; **read/write** access; type: `string` or `nil`. Global identifier for use in distributed simulation or `NIL` if the link is not distributed.

goal_latency

Optional attribute; **read/write** access; type: `float`. Goal latency in seconds for this link. See also the `set-min-latency` command.

Command List

Commands

connect-real-network-bridge	connect to the real network
connect-real-network-host	connect to the real network
connect-real-network-napt	enable NAPT from simulated network
connect-real-network-router	<i>deprecated</i> — connect to the real network
disconnect-real-network	disconnect from the real network
get-free-connector	return the name of an unused connector
info	print information about the device
pcap-dump	Dump network traffic to a file, in libpcap format
pcap-dump-stop	stop the current dump
set-goal-latency	Set the link's goal latency (in seconds).
status	print status of the device
tcpdump	run the tcpdump program
tcpdump-stop	stop the current tcpdump capture
wireshark	run the wireshark/ethereal program
wireshark-stop	stop the current wireshark capture

Command Descriptions

<ethernet_hub>.connect-real-network-bridge

Synopsis

```
<ethernet_hub>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
<ethernet_link>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
<ethernet_cable>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
<ethernet_switch>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
```

Description

Creates an Ethernet bridge between a simulated Ethernet link and a real network through an Ethernet interface of the simulation host.

The optional *interface* argument specifies the Ethernet or TAP interface of the host to use.

By default a TAP interface is used, but if the *host-access* argument is `raw`, raw access to an Ethernet interface is used.

MAC address translation can be disabled with the `-no-mac-xlate` flag.

The `-persistent` is for backward compatibility and should not be used.

If `-propagate-link-status` is specified, link status changes on the host interface will be propagated to all devices on the link that implements the link-status interface. For TAP, only 'up' and 'down' status will be propagated (and not 'unconnected'). Link status propagation is only supported on Linux.

This command returns the new real-network component object.

See the *Connecting to a Real Network* chapter of the *Ethernet Networks in Simics* manual for more information about how to connect to a real network.

See Also

[connect-real-network](#), [connect-real-network-host](#), [disconnect-real-network](#)

<ethernet_hub>.connect-real-network-host

Synopsis

```
<ethernet_hub>.connect-real-network-host ["interface"] [-persistent]
<ethernet_link>.connect-real-network-host ["interface"] [-persistent]
```

```
<ethernet_cable>.connect-real-network-host ["interface"] [-persistent]
<ethernet_switch>.connect-real-network-host ["interface"] [-persistent]
connect-real-network-host ["interface"] [-persistent]
```

Description

Connects a TAP interface of the simulation host to a simulated Ethernet link.

The optional *interface* argument specifies the TAP interface of the host to use.

The *-persistent* is for backward compatibility and should not be used.

This command returns the new real-network component object.

See the *Connecting to a Real Network* chapter of the *Simics User Guide* for more information about how to connect to a real network.

See Also

[connect-real-network](#), [connect-real-network-bridge](#), [disconnect-real-network](#)

```
<ethernet_hub>.connect-real-network-napt
```

Synopsis

```
<ethernet_hub>.connect-real-network-napt [service-node]
<ethernet_link>.connect-real-network-napt [service-node]
<ethernet_cable>.connect-real-network-napt [service-node]
<ethernet_switch>.connect-real-network-napt [service-node]
<ethernet_vlan_switch>.connect-real-network-napt [service-node]
connect-real-network-napt ethernet-link [service-node]
```

Description

Enables machines on the simulated network to initiate accesses to real hosts without the need to configure the simulated machine with a real IP address. NAPT (Network Address Port Translation) uses the IP address and a port number of the host that Simics is running on to perform the access. Replies are then translated back to match the request from the simulated machine. This command also enables NAPT for accesses that are initiated from the simulated machine.

See Also

[connect-real-network](#), [connect-real-network-port-in](#), [connect-real-network-port-out](#), [connect-real-network-host](#), [connect-real-network-bridge](#)

```
<ethernet_hub>.connect-real-network-router — deprecated
```

Synopsis

```
<ethernet_hub>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]
<ethernet_link>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]
<ethernet_cable>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]
```

```
<ethernet_switch>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]
connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]
```

Description

This command is deprecated; use [connect-real-network-bridge](#) or [connect-real-network-host](#) instead.

Deprecated command

See Also

[connect-real-network](#), [connect-real-network-host](#), [connect-real-network-bridge](#), [disconnect-real-network](#)

```
<ethernet_hub>.disconnect-real-network
```

Synopsis

```
<ethernet_hub>.disconnect-real-network
<ethernet_link>.disconnect-real-network
<ethernet_cable>.disconnect-real-network
<ethernet_switch>.disconnect-real-network
disconnect-real-network
```

Description

Closes all connections to real networks except port forwarding and NAPT.

See Also

[connect-real-network-host](#), [connect-real-network-bridge](#)

```
<ethernet_hub>.get-free-connector
```

Synopsis

```
<ethernet_hub>.get-free-connector
```

Description

This command returns the name of a connector which is not connected to anything.

```
<ethernet_hub>.info
```

Synopsis

```
<ethernet_hub>.info
```

Description

Print detailed information about the configuration of the device.

```
<ethernet_hub>.pcap-dump
```

Synopsis

```
<ethernet_hub>.pcap-dump file
<ethernet_hub>.pcap-dump-stop
```

Description

```
<ethernet_hub>.pcap-dump-stop
```

Synopsis

```
<ethernet_hub>.pcap-dump-stop
<ethernet_hub>.pcap-dump file
```

Description

```
<ethernet_hub>.set-goal-latency
```

Synopsis

```
<ethernet_hub>.set-goal-latency latency
```

Description

```
<ethernet_hub>.status
```

Synopsis

```
<ethernet_hub>.status
```

Description

Print detailed information about the current status of the device.

```
<ethernet_hub>.tcpdump
```

Synopsis

```
<ethernet_hub>.tcpdump [“flags”]
<ethernet_hub>.tcpdump-stop
```

Description

Runs the **tcpdump** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to **tcpdump**.

```
<ethernet_hub>.tcpdump-stop
```

Synopsis

```
<ethernet_hub>.tcpdump-stop  
<ethernet_hub>.tcpdump ["flags"]
```

Description

Runs the **tcpdump** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to **tcpdump**.

<ethernet_hub>.wireshark

Alias

```
<ethernet_hub>.ethereal
```

Synopsis

```
<ethernet_hub>.wireshark ["flags"]  
<ethernet_hub>.wireshark-stop
```

Description

Runs the **wireshark** or **ethereal** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to program.

<ethernet_hub>.wireshark-stop

Alias

```
<ethernet_hub>.ethereal-stop
```

Synopsis

```
<ethernet_hub>.wireshark-stop  
<ethernet_hub>.wireshark ["flags"]
```

Description

Runs the **wireshark** or **ethereal** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to program.

ethernet_switch

Provided By
[eth-links](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

Ethernet switch: this component represents a switched ethernet network, allowing any number of devices to connect and optimizing the packet routing according to what is learned about the MAC addresses talking on the link.

Connectors

Name	Type	Direction
device0	ethernet-link	any

Attributes

global_id

Optional attribute; **read/write** access; type: `string` or `nil`. Global identifier for use in distributed simulation or `NIL` if the link is not distributed.

goal_latency

Optional attribute; **read/write** access; type: `float`. Goal latency in seconds for this link. See also the `set-min-latency` command.

Command List

Commands

connect-real-network-bridge	connect to the real network
connect-real-network-host	connect to the real network
connect-real-network-napt	enable NAPT from simulated network
connect-real-network-router	<i>deprecated</i> — connect to the real network
disconnect-real-network	disconnect from the real network
get-free-connector	return the name of an unused connector
info	print information about the device
pcap-dump	Dump network traffic to a file, in libpcap format
pcap-dump-stop	stop the current dump
set-goal-latency	Set the link's goal latency (in seconds).
status	print status of the device
tcpdump	run the tcpdump program
tcpdump-stop	stop the current tcpdump capture
wireshark	run the wireshark/ethereal program

wireshark-stop stop the current wireshark capture

Command Descriptions

<ethernet_switch>.connect-real-network-bridge

Synopsis

```
<ethernet_switch>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
<ethernet_link>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
<ethernet_cable>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
<ethernet_hub>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
```

Description

Creates an Ethernet bridge between a simulated Ethernet link and a real network through an Ethernet interface of the simulation host.

The optional *interface* argument specifies the Ethernet or TAP interface of the host to use.

By default a TAP interface is used, but if the *host-access* argument is `raw`, raw access to an Ethernet interface is used.

MAC address translation can be disabled with the `-no-mac-xlate` flag.

The `-persistent` is for backward compatibility and should not be used.

If `-propagate-link-status` is specified, link status changes on the host interface will be propagated to all devices on the link that implements the link-status interface. For TAP, only 'up' and 'down' status will be propagated (and not 'unconnected'). Link status propagation is only supported on Linux.

This command returns the new real-network component object.

See the *Connecting to a Real Network* chapter of the *Ethernet Networks in Simics* manual for more information about how to connect to a real network.

See Also

[connect-real-network](#), [connect-real-network-host](#), [disconnect-real-network](#)

<ethernet_switch>.connect-real-network-host

Synopsis

```
<ethernet_switch>.connect-real-network-host ["interface"] [-persistent]
<ethernet_link>.connect-real-network-host ["interface"] [-persistent]
```

```
<ethernet_cable>.connect-real-network-host ["interface"] [-persistent]
<ethernet_hub>.connect-real-network-host ["interface"] [-persistent]
connect-real-network-host ["interface"] [-persistent]
```

Description

Connects a TAP interface of the simulation host to a simulated Ethernet link.

The optional *interface* argument specifies the TAP interface of the host to use.

The *-persistent* is for backward compatibility and should not be used.

This command returns the new real-network component object.

See the *Connecting to a Real Network* chapter of the *Simics User Guide* for more information about how to connect to a real network.

See Also

[connect-real-network](#), [connect-real-network-bridge](#), [disconnect-real-network](#)

```
<ethernet_switch>.connect-real-network-napt
```

Synopsis

```
<ethernet_switch>.connect-real-network-napt [service-node]
<ethernet_link>.connect-real-network-napt [service-node]
<ethernet_cable>.connect-real-network-napt [service-node]
<ethernet_hub>.connect-real-network-napt [service-node]
<ethernet_vlan_switch>.connect-real-network-napt [service-node]
connect-real-network-napt ethernet-link [service-node]
```

Description

Enables machines on the simulated network to initiate accesses to real hosts without the need to configure the simulated machine with a real IP address. NAPT (Network Address Port Translation) uses the IP address and a port number of the host that Simics is running on to perform the access. Replies are then translated back to match the request from the simulated machine. This command also enables NAPT for accesses that are initiated from the simulated machine.

See Also

[connect-real-network](#), [connect-real-network-port-in](#), [connect-real-network-port-out](#), [connect-real-network-host](#), [connect-real-network-bridge](#)

```
<ethernet_switch>.connect-real-network-router — deprecated
```

Synopsis

```
<ethernet_switch>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]
<ethernet_link>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]
<ethernet_cable>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]
```

```
<ethernet_hub>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]

connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]
```

Description

This command is deprecated; use [connect-real-network-bridge](#) or [connect-real-network-host](#) instead.

Deprecated command

See Also

[connect-real-network](#), [connect-real-network-host](#), [connect-real-network-bridge](#), [disconnect-real-network](#)

```
<ethernet_switch>.disconnect-real-network
```

Synopsis

```
<ethernet_switch>.disconnect-real-network
<ethernet_link>.disconnect-real-network
<ethernet_cable>.disconnect-real-network
<ethernet_hub>.disconnect-real-network
disconnect-real-network
```

Description

Closes all connections to real networks except port forwarding and NAPT.

See Also

[connect-real-network-host](#), [connect-real-network-bridge](#)

```
<ethernet_switch>.get-free-connector
```

Synopsis

```
<ethernet_switch>.get-free-connector
```

Description

This command returns the name of a connector which is not connected to anything.

```
<ethernet_switch>.info
```

Synopsis

```
<ethernet_switch>.info
```

Description

Print detailed information about the configuration of the device.

```
<ethernet_switch>.pcap-dump
```

Synopsis

```
<ethernet_switch>.pcap-dump file
<ethernet_switch>.pcap-dump-stop
```

Description

```
<ethernet_switch>.pcap-dump-stop
```

Synopsis

```
<ethernet_switch>.pcap-dump-stop
<ethernet_switch>.pcap-dump file
```

Description

```
<ethernet_switch>.set-goal-latency
```

Synopsis

```
<ethernet_switch>.set-goal-latency latency
```

Description

```
<ethernet_switch>.status
```

Synopsis

```
<ethernet_switch>.status
```

Description

Print detailed information about the current status of the device.

```
<ethernet_switch>.tcpdump
```

Synopsis

```
<ethernet_switch>.tcpdump [“flags”]
<ethernet_switch>.tcpdump-stop
```

Description

Runs the **tcpdump** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to **tcpdump**.

```
<ethernet_switch>.tcpdump-stop
```

Synopsis

```
<ethernet_switch>.tcpdump-stop  
<ethernet_switch>.tcpdump ["flags"]
```

Description

Runs the **tcpdump** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to **tcpdump**.

<ethernet_switch>.wireshark

Alias

```
<ethernet_switch>.ethereal
```

Synopsis

```
<ethernet_switch>.wireshark ["flags"]  
<ethernet_switch>.wireshark-stop
```

Description

Runs the **wireshark** or **ethereal** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to program.

<ethernet_switch>.wireshark-stop

Alias

```
<ethernet_switch>.ethereal-stop
```

Synopsis

```
<ethernet_switch>.wireshark-stop  
<ethernet_switch>.wireshark ["flags"]
```

Description

Runs the **wireshark** or **ethereal** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to program.

ethernet_vlan_switch

Provided By
[eth-links](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

Ethernet VLAN switch: this component represents a switched ethernet network with VLAN support. Any number of devices is allowed to connect to various ports of the switch. Each port can be configured with its own VLAN information, in order to create sub-networks in the switch.

Connectors

Name	Type	Direction
trunk_dev0	ethernet-link	any

Attributes

global_id

Optional attribute; **read/write** access; type: `string` or `nil`. Global identifier for use in distributed simulation or `NIL` if the link is not distributed.

goal_latency

Optional attribute; **read/write** access; type: `float`. Goal latency in seconds for this link. See also the `set-min-latency` command.

Command List

Commands

add-vlan	add a VLAN definition and corresponding connectors
connect-real-network-napt	enable NAPT from simulated network
get-free-connector	return the name of an unused connector
get-free-trunk-connector	return the name of an unused trunk connector
info	print information about the device
pcap-dump	Dump network traffic to a file, in libpcap format
pcap-dump-stop	stop the current dump
set-goal-latency	Set the link's goal latency (in seconds).
status	print status of the device
tcpdump	run the tcpdump program
tcpdump-stop	stop the current tcpdump capture
wireshark	run the wireshark/ethereal program
wireshark-stop	stop the current wireshark capture

Command Descriptions

`<ethernet_vlan_switch>.add-vlan`

Synopsis

`<ethernet_vlan_switch>.add-vlan vlan_id`

Description

`<ethernet_vlan_switch>.connect-real-network-napt`

Synopsis

`<ethernet_vlan_switch>.connect-real-network-napt [service-node]
<ethernet-link>.connect-real-network-napt [service-node]
<ethernet_cable>.connect-real-network-napt [service-node]
<ethernet_hub>.connect-real-network-napt [service-node]
<ethernet_switch>.connect-real-network-napt [service-node]
connect-real-network-napt ethernet-link [service-node]`

Description

Enables machines on the simulated network to initiate accesses to real hosts without the need to configure the simulated machine with a real IP address. NAPT (Network Address Port Translation) uses the IP address and a port number of the host that Simics is running on to perform the access. Replies are then translated back to match the request from the simulated machine. This command also enables NAPT for accesses that are initiated from the simulated machine.

See Also

[connect-real-network](#), [connect-real-network-port-in](#), [connect-real-network-port-out](#),
[connect-real-network-host](#), [connect-real-network-bridge](#)

`<ethernet_vlan_switch>.get-free-connector`

Synopsis

`<ethernet_vlan_switch>.get-free-connector vlan_id`

Description

This command returns the name of a connector which is not connected to anything.

`<ethernet_vlan_switch>.get-free-trunk-connector`

Synopsis

```
<ethernet_vlan_switch>.get-free-trunk-connector [vlan_id]
```

Description

This command returns the name of a connector which is not connected to anything.

```
<ethernet_vlan_switch>.info
```

Synopsis

```
<ethernet_vlan_switch>.info
```

Description

Print detailed information about the configuration of the device.

```
<ethernet_vlan_switch>.pcap-dump
```

Synopsis

```
<ethernet_vlan_switch>.pcap-dump file  
<ethernet_vlan_switch>.pcap-dump-stop
```

Description

```
<ethernet_vlan_switch>.pcap-dump-stop
```

Synopsis

```
<ethernet_vlan_switch>.pcap-dump-stop  
<ethernet_vlan_switch>.pcap-dump file
```

Description

```
<ethernet_vlan_switch>.set-goal-latency
```

Synopsis

```
<ethernet_vlan_switch>.set-goal-latency latency
```

Description

```
<ethernet_vlan_switch>.status
```

Synopsis

```
<ethernet_vlan_switch>.status
```

Description

Print detailed information about the current status of the device.

```
<ethernet_vlan_switch>.tcpdump
```

Synopsis

```
<ethernet_vlan_switch>.tcpdump ["flags"]  
<ethernet_vlan_switch>.tcpdump-stop
```

Description

Runs the **tcpdump** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to **tcpdump**.

```
<ethernet_vlan_switch>.tcpdump-stop
```

Synopsis

```
<ethernet_vlan_switch>.tcpdump-stop  
<ethernet_vlan_switch>.tcpdump ["flags"]
```

Description

Runs the **tcpdump** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to **tcpdump**.

```
<ethernet_vlan_switch>.wireshark
```

Alias

```
<ethernet_vlan_switch>.ethereal
```

Synopsis

```
<ethernet_vlan_switch>.wireshark ["flags"]  
<ethernet_vlan_switch>.wireshark-stop
```

Description

Runs the **wireshark** or **ethereal** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to program.

```
<ethernet_vlan_switch>.wireshark-stop
```

Alias

```
<ethernet_vlan_switch>.ethereal-stop
```

Synopsis

```
<ethernet_vlan_switch>.wireshark-stop  
<ethernet_vlan_switch>.wireshark ["flags"]
```

Description

Runs the **wireshark** or **ethereal** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to program.

example-keypad

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

Example of using the status_panel as a simple keypad.

Connectors

Name	Type	Direction
device	keypad	up

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<example-keypad>.info

Synopsis

<example-keypad>.info

Description

Print detailed information about the configuration of the device.

<example-keypad>.status

Synopsis

<example-keypad>.status

Description

Print detailed information about the current status of the device.

example-status-panel

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description
Example use of the status_panel class.

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

[info](#) print information about the device
[status](#) print status of the device

Command Descriptions

<example-status-panel>.info

Synopsis

<example-status-panel>.info

Description

Print detailed information about the configuration of the device.

<example-status-panel>.status

Synopsis

<example-status-panel>.status

Description

Print detailed information about the current status of the device.

generic_pcie_switch

Provided By

[generic-pcie-switch-comp](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

Generic PCIe switch with a configurable number of ports and configurable vendor and device IDs

Connectors

Name	Type	Direction
up	pci-bus	up
down[0]	pci-bus	down
down[1]	pci-bus	down
down[2]	pci-bus	down
down[3]	pci-bus	down
down[4]	pci-bus	down

Command List

Commands

[info](#) print information about the device
[status](#) print status of the device

Command Descriptions

<generic_pcie_switch>.info

Synopsis

<generic_pcie_switch>.info

Description

Print detailed information about the configuration of the device.

<generic_pcie_switch>.status

Synopsis

<generic_pcie_switch>.status

Description

Print detailed information about the current status of the device.

i2c_link_v2

Provided By
[i2c-link-v2](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

This component represents a simple i2c link allowing any number of devices to connect.

Connectors

Name	Type	Direction
device0	i2c-link	any

Attributes

global_id

Optional attribute; **read/write** access; type: `string` or `nil`. Global identifier for use in distributed simulation or `NIL` if the link is not distributed.

goal_latency

Optional attribute; **read/write** access; type: `float`. Goal latency in seconds for this link. See also the `set-min-latency` command.

Command List

Commands

[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

<i2c_link_v2>.info

Synopsis

<i2c_link_v2>.info

Description

Print detailed information about the configuration of the device.

<i2c_link_v2>.status

Synopsis

<i2c_link_v2>.status

Description

Print detailed information about the current status of the device.

instruction-data-splitter

Provided By
[timing-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description
Splits memory transactions into separate instruction and data branches

Connectors

Name	Type	Direction
prev-level	cache	up
dbranch	cache	down
ibranch	cache	down

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<instruction-data-splitter>.info

Synopsis

<instruction-data-splitter>.info

Description

Print detailed information about the configuration of the device.

<instruction-data-splitter>.status

Synopsis

<instruction-data-splitter>.status

Description

Print detailed information about the current status of the device.

isa-fourport

Provided By
[isa-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

This component represents a fourport ISA card with four NS16550 serial ports. To access the serial ports in Linux, use “setserial /dev/ttys5 port 0x2a0 auto_irq autoconfig”

Connectors

Name	Type	Direction
isa-bus	isa-bus	up
com[1-4]	serial	down

Attributes

base_port

Optional attribute; **read/write** access; type: `integer`. The starting port number in I/O space. The default port is 0x2a0, and the mapping is 32 bytes large. Allowed base ports are 0x2a0 and 0x1a0.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

irq_level

Optional attribute; **read/write** access; type: `integer`. The interrupt level for the fourport device, default is 2.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

`<isa-fourport>.info`

Synopsis

`<isa-fourport>.info`

Description

Print detailed information about the configuration of the device.

`<isa-fourport>.status`

Synopsis

`<isa-fourport>.status`

Description

Print detailed information about the current status of the device.

isa-lance

Provided By
[isa-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “isa-lance” component represents an ISA bus based Ethernet adapter.

Connectors

Name	Type	Direction
isa-bus	isa-bus	up
ethernet	ethernet-link	down

Attributes

base_port

Optional attribute; **read/write** access; type: `integer`. The starting port number in I/O space. The default port is 0x300, and the mapping is 0x17 bytes large.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

irq_level

Optional attribute; **read/write** access; type: `integer`. The interrupt level for the Lance device, default is 7.

mac_address

Required attribute; **read/write** access; type: `string`. The MAC address of the Ethernet adapter.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<`isa-lance`>.info

Synopsis

<`isa-lance`>.info

Description

Print detailed information about the configuration of the device.

<`isa-lance`>.status

Synopsis

<`isa-lance`>.status

Description

Print detailed information about the current status of the device.

isa-vga

Provided By

[isa-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “isa-vga” component represents an ISA bus based VGA compatible graphics adapter.

Connectors

Name	Type	Direction
isa-bus	isa-bus	up
console	graphics-console	down

Attributes

bios

Optional attribute; **read/write** access; type: `string`. The VGA BIOS file to use.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<isa-vga>.info

Synopsis

<isa-vga>.info

Description

Print detailed information about the configuration of the device.

<isa-vga>.status

Synopsis

<isa-vga>.status

Description

Print detailed information about the current status of the device.

memory-timer

Provided By
[timing-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description
Timing model of memory

Connectors

Name	Type	Direction
prev-level	cache	up

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<memory-timer>.info

Synopsis

<memory-timer>.info

Description

Print detailed information about the configuration of the device.

<memory-timer>.status

Synopsis

<memory-timer>.status

Description

Print detailed information about the current status of the device.

micron_mtfc2ggqdi_emmc_card

Provided By

[mmc-card-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

A 2GB Micron eMMC card.

Connectors

Name	Type	Direction
mmc_controller	mmc	up

Command List

Commands

[info](#) print information about the device
[status](#) print status of the device

Command Descriptions

`<micron_mtfc2ggqdi_emmc_card>.info`

Synopsis

`<micron_mtfc2ggqdi_emmc_card>.info`

Description

Print detailed information about the configuration of the device.

`<micron_mtfc2ggqdi_emmc_card>.status`

Synopsis

`<micron_mtfc2ggqdi_emmc_card>.status`

Description

Print detailed information about the current status of the device.

micron_mtfc4ggqdi_emmc_card

Provided By

[mmc-card-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

A Micron MTFC4GGQDI-IT eMMC Card.

Connectors

Name	Type	Direction
mmc_controller	mmc	up

Command List

Commands

[info](#) print information about the device
[status](#) print status of the device

Command Descriptions

<micron_mtfc4ggqdi_emmc_card>.info

Synopsis

[<micron_mtfc4ggqdi_emmc_card>.info](#)

Description

Print detailed information about the configuration of the device.

<micron_mtfc4ggqdi_emmc_card>.status

Synopsis

[<micron_mtfc4ggqdi_emmc_card>.status](#)

Description

Print detailed information about the current status of the device.

micron_mtfc4ggqdi_sdhc_card

Provided By

[mmc-card-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

A SDHC card component similar to Micron MTFC4GGQDI-IT eMMC.

Connectors

Name	Type	Direction
mmc_controller	mmc	up

Command List

Commands

[info](#) print information about the device
[status](#) print status of the device

Command Descriptions

`<micron_mtfc4ggqdi_sdhc_card>.info`

Synopsis

`<micron_mtfc4ggqdi_sdhc_card>.info`

Description

Print detailed information about the configuration of the device.

`<micron_mtfc4ggqdi_sdhc_card>.status`

Synopsis

`<micron_mtfc4ggqdi_sdhc_card>.status`

Description

Print detailed information about the current status of the device.

os_awareness

Provided By

[os-awareness](#)

Interfaces Implemented

[component](#), [conf_object](#), [log_object](#)

Description

The “os-awareness” component tracks software running on the system.

Command List

Commands

active-node	list the active nodes
break-enter	break the simulation when a processor becomes active on the node
break-exit	break the simulation when a processor becomes active on the node
code-coverage	collect a coverage profile
disable-tracker	stop using software tracking
enable-tracker	enable software tracking
find	find a node
info	print information about the device
linux-autodetect-settings	Try to autodetect settings for the Linux tracker
list	list processes/tasks
load-parameters	Load tracker parameter settings
log-syscalls	start logging system calls
node-info	show software tracker node information
node-tree	list software node tree
ose-detect-settings	detect settings for the OSE tracker
partition-settings	configure the partition tracker
qnx-detect-settings	detect settings for the QNX tracker
status	print status of the device
track	let a context track a process, thread, task, etc.
unbreak	cancel a software breakpoint
vxworks-detect-settings	detect settings for the VxWorks tracker
wait-for-activated	Wait for a processor to become activated on the node <i>deprecated</i> —
wait-for-active	Wait for a processor to become deactivated on the node <i>deprecated</i> —
wait-for-deactivated	detect settings for the Wind River hypervisor tracker
wait-for-inactive	
wr-hypervisor-detect-settings	

Command Descriptions

<os_awareness>.active-node

Synopsis

<os_awareness>.active-node [cpu-name|-all]

Description

List the leaf nodes that have an active processor.

If the optional *cpu* parameter is given, only the active node for this processor is printed. The canonical node path for that processor is also returned.

Without any arguments the active leaf node for all processors that the os awareness system is aware of will be listed.

`<os_awareness>.break-enter`

Synopsis

```
<os_awareness>.break-enter node [-once] [-immediately]
```

Description

Add a breakpoint that will stop the simulation when a processor known by the tracker system becomes active on a node matching the node path pattern.

The *node* parameter is a *node path pattern* that identifies the software node to break on. See the *Analyzer User's Guide* for more information about node path patterns. The node path pattern will be evaluated every time a node becomes active in order to check if the break condition is fulfilled. This means that a node that did not exist when this command was issued might cause the simulation to halt. There is also a performance penalty involved. If only one node is of interest, a node id can be given (as an integer value) instead.

If the *-once* flag is specified, the breakpoint will be removed automatically when it is triggered. Also, if a node matching the node path is already active and the *-immediately* flag is specified, no breakpoint will be set.

If the *-immediately* flag is specified and any processor is active on any of the nodes matching the node path pattern, this command will break directly.

The return value is a breakpoint ID; give it to `<os_awareness>.unbreak` to delete the breakpoint.

See Also

[`<os_awareness>.break-exit`](#), [`<os_awareness>.wait-for-enter`](#), [`<os_awareness>.wait-for-exit`](#), [`<os_awareness>.unbreak`](#)

`<os_awareness>.break-exit`

Synopsis

```
<os_awareness>.break-exit node [-once] [-immediately]
```

Description

Add a breakpoint that will stop the simulation when a processor known by the tracker system becomes inactive on a node matching the node path pattern.

The *node* parameter is a *node path pattern* that identifies the software nodes that should be inactive before breaking. See the *Analyzer User's Guide* for more information about

node path patterns. The node path pattern will be evaluated every time a node becomes inactive in order to check if the break condition is fulfilled. This means that a node that did not exist when this command was issued might cause the simulation to continue. There is also a performance penalty involved. If only one node is of interest, a node id can be given (as an integer value) instead.

If the *-once* flag is specified, the breakpoint will be removed automatically when it is triggered. Also, if no nodes matching the node path are active and the *-immediately* flag is specified, no breakpoint will be set.

If the *-immediately* flag is specified and no processor is active on any of the nodes matching the node path pattern, this command will break directly.

The return value is a breakpoint ID; give it to `<os_awareness>.unbreak` to delete the breakpoint.

See Also

`<os_awareness>.break-enter`, `<os_awareness>.wait-for-enter`, `<os_awareness>.wait-for-exit`, `<os_awareness>.unbreak`

`<os_awareness>.code-coverage`

Synopsis

`<os_awareness>.code-coverage [node] (binary|syntable) [output] [format] [-next|-running]`

Description

Collect coverage data for a process running on the system.

This command creates a **coverage_profiler** object that will collect a code coverage profile of the process identified by *node*, a *node path pattern* (see the *Analyzer User's Guide* for more information).

If *output* is specified, the resulting profile will be saved automatically when the process finished. The result is saved to *output* and the output format is selected with *format*. The default format is `html`, which will create a directory of HTML files. The `raw` format is useful for postprocessing to create other formats by external means.

The *output* is not specified, the output can be saved using the `<output_profiler>.save` command.

To create the coverage profile, the command will load debug information from *binary* or use existing information from *syntable* and use it to determine which source lines have been executed. The binary must be identical to the binary that is run on the target system. If no *node* is given, the last part of *binary* will be used.

Using a **syntable** object allows greater flexibility, in particular since it supports loading debug information from several binaries which is needed when the main binary calls functions in shared libraries. And it supports defining load addresses for every binary which is required when dynamically loaded libraries are relocated or when binaries are explicitly relocated, for example by a boot loader.

If the source files are not located where the debug information in the binary points, use a **syhtable** object and use the **<syhtable>.source-path** command to provide source path translations to find the sources.

The return value is the **coverage_profiler** object.

See Also

[`<coverage_profiler>.save`](#)

`<os_awareness>.disable-tracker`

Synopsis

`<os_awareness>.disable-tracker`

Description

Stop tracking the software running on the target system.

When tracking is disabled, it is no longer possible to follow individual processes and tasks. Note that if another user, such as Eclipse, has activated the tracker, the tracker will remain active.

See Also

[`<os_awareness>.enable-tracker`](#)

`<os_awareness>.enable-tracker`

Synopsis

`<os_awareness>.enable-tracker`

Description

Start tracking the software running on the target system.

This allows inspection of processes and tasks and other operating system information for the software running on the target. It is also required to do other tracking operations such as process debugging or analysis.

See Also

[`<os_awareness>.disable-tracker`](#)

`<os_awareness>.find`

Synopsis

`<os_awareness>.find "node" [-unique] [-raise] [-id]`

Description

Search for a node in the node tree. The find command takes a node specification as input and from that finds all nodes matching the expression. By default the matching nodes are represented with their node path specification. If this command is run from

the CLI the matching nodes will be printed on the console. If this command is run from a script it will return a list with the result.

The *node* contains the node specification to use when searching for matching nodes. This is backward compatible with the old system. If the *node* argument is an integer only, it will be treated as a node id. For more information about node specifications please see the *Analyzer User's Guide*.

The *id* parameter can be specified in order to identify the matched nodes with their node id instead of with a node path.

The *unique* parameter can be specified to only allow one matching node to be found. If multiple matches are found, no matches at all will be listed.

The *raise* can be used to specify that the command should raise a CliError if no nodes are found. It can also be used in combination with the *unique* parameter to force an exception if multiple nodes matches the node specification.

<os_awareness>.info

Synopsis

```
<os_awareness>.info
```

Description

Print detailed information about the configuration of the device.

<os_awareness>.linux-autodetect-settings

Synopsis

```
<os_awareness>.linux-autodetect-settings [param-file] [node] ["version-string"] [base-address] [symbol-file] [-overwrite]
```

Description

Autodetect the parameters to use with the Linux tracker. For this to work, Linux must be already up and running on the simulated machine. In case the Linux instance does not own the entire machine (for instance, it might be a hypervisor guest). The optional *node* argument can be used to specify on which node Linux is running, this is useful if Linux runs inside a hypervisor. If no node is specified the root node will be used. The optional *param-file* argument is used to specify where to save the parameters, the default is 'autodetect'. The optional *version-string* argument can be used to specify information about the Linux version running on the system. This will for example be visible in the node-tree and from Eclipse. The default is an empty string. The optional *base-address* argument can be used to specify the kernel base address. This argument is optional but can improve the detection time.

The optional *symbol-file* argument can be used to point out the kernel symbol file (System.map or vmlinuz). This is required for newer x86 kernels (2.6.32 and higher), and can be left out for other kernels. If it is given but no useful information can be extracted from it, it will be ignored. If it is not already available, the System.map file can be generated on the target system: with /proc mounted, run cat /proc/kallsyms > System.map.

See Also

[`<os_awareness>.load-parameters`](#)

<os_awareness>.list

Synopsis

`<os_awareness>.list`

Description

<os_awareness>.load-parameters

Synopsis

`<os_awareness>.load-parameters` *file*

Description

Load configuration parameters for a software tracker from a file. The *file* argument should point to the file containing the parameters configuration.

See Also

[`<os_awareness>.linux-autodetect-settings`](#), [`<os_awareness>.qnx-detect-settings`](#), [`<os_awareness>.ose-detect-settings`](#), [`<os_awareness>.vxworks-detect-settings`](#)

<os_awareness>.log-syscalls

Synopsis

`<os_awareness>.log-syscalls` [-disable]

Description

<os_awareness>.node-info

Synopsis

`<os_awareness>.node-info` *node*

Description

<os_awareness>.node-tree

Synopsis

```
<os_awareness>.node-tree [-id]
```

Description

Lists the node tree for an active software tracker for debugging purposes. In the general case the first level node represents an operating system. The second level nodes represent the kernel and the userspace of the OS. Further node levels are tracker specific.

The *id* argument specifies if node ids should be printed in the node tree.

See document “Analyzer - User’s Guide”, chapter “OS Awareness Details”.

Note

Intended for debugging

<os_awareness>.ose-detect-settings

Synopsis

```
<os_awareness>.ose-detect-settings symbol-file [param-file] ["version-string"] [ose-version]  
[-overwrite]
```

Description

This command generates a parameter file for an OSE configuration. It takes the path to a file containing symbols for the OSE system, given by the *file* argument. The resulting parameter file will be written to *output* or ‘ose.params’ if *output* is left out. The optional *ose-version* argument can be used to specify the OSE version. This has the benefit of allowing the parameter detection on a system that is not yet booted. The *overwrite* argument can be used in order to allow overwrite of an old file with new parameters.

See Also

[`<os_awareness>.load-parameters`](#)

<os_awareness>.partition-settings

Synopsis

```
<os_awareness>.partition-settings [param-file] [node] ["version-string"] [-overwrite]  
["partition-name"] [processor] [-all] [-update]
```

Description

Configure the parameters to use with the partition tracker. This tracker can, for example, be used to collect coverage on a system for which there is no native tracker available. See the *Analyzer User’s Guide* for more information.

The optional *node* argument can be used to specify on which node to attach the partition tracker. If no node is specified the root node will be used. The optional *param-file* argument is used to specify where to save the parameters, the default is ‘partition.params’. The optional *version-string* argument can be used to specify the name of the partition trackers root node, the default is ‘Partition’.

The optional *partition-name* argument can be used to specify which partition that should be created or updated, the default is 'Partition'.

The optional *processor* argument can be used to specify the processor that should be added to the partition. By default no processor is added. Notice that this argument can not be used in combination with the *-all* argument.

The optional *-all* argument can be used to specify that all available processors should be added to the partition. Notice that this argument can not be used in combination with the *processors* argument.

The optional *-overwrite* argument can be used to allow overwriting an already existing file.

The optional *-update* argument can be used to specify that an existing parameters file should be updated. This is useful if, for example, a second partition should be added to the configuration.

See Also

[`<os_awareness>.load-parameters`](#)

`<os_awareness>.qnx-detect-settings`

Synopsis

`<os_awareness>.qnx-detect-settings symbol-file [param-file] ["version-string"] [-overwrite]`

Description

This command generates a parameter file for a QNX configuration. It takes the path to a file containing symbols for the QNX system, given by the *file* argument. The resulting parameter file will be written to *output* or 'qnx.params' if *output* is left out. The *overwrite* argument can be used in order to allow overwrite of an old file with new parameters.

See Also

[`<os_awareness>.load-parameters`](#)

`<os_awareness>.status`

Synopsis

`<os_awareness>.status`

Description

Print detailed information about the current status of the device.

`<os_awareness>.track`

Synopsis

`<os_awareness>.track node [syntable] [-next|-running] ["context-name"]`

Description

Track a process (or thread, task, etc.) by making sure that the corresponding context object is active when the process is running on a processor.

The *node* parameter determines the process to track. It is an *node path pattern* that is used to match against the software nodes. This can be the process name or other properties such as pid. It is possible to track not only processes, but any software node including OS kernels and individual threads. See the *Analyzer User's Guide* for more information about node path patterns.

The *symtable* parameter can be used to specify an existing **symtable** object instead of letting the command create a new one.

This command will look for a running process that matches the node path pattern and attach to it. If no such process is found, it will wait for a new process to start and attach to it then. The *-next* flag forces it to ignore already running processes, and the *-running* flag makes it only consider running processes and fail if none were found.

The optional *context-name* argument can be specified in order to give the created context object a specific name.

The new **context** object is returned from this command.

<os_awareness>.unbreak

Synopsis

```
<os_awareness>.unbreak ("bp_id"|-all)
```

Description

Remove a breakpoint that was created with **<os_awareness>.break-enter** or **<os_awareness>.break-exit**. These commands returns an id that is used to identify which breakpoint to remove.

This id should be given as the *bp_id* parameter.

See Also

[**<os_awareness>.break-enter**](#), [**<os_awareness>.break-exit**](#)

<os_awareness>.vxworks-detect-settings

Synopsis

```
<os_awareness>.vxworks-detect-settings symbol-file [param-file] [node] ["version-string"]  
[vxworks-version] [-overwrite]
```

Description

This command generates a parameter file for a VxWorks configuration. It takes the path to a file containing symbols for the VxWorks system, given by the *file* argument. The resulting parameter file will be written to *output* or 'vxworks.params' if *output* is left out. The optional *version-string* argument can be used to specify information about the VxWorks version running on the system. This will for example be visible in the node-tree and from Eclipse. It default to the empty string. The optional *vxworks-version*

argument can be used to specify the VxWorks version. This has the benefit of allowing the parameter detection on a system that is not yet booted. The *overwrite* argument can be used in order to allow overwrite of an old file with new parameters.

See Also

[`<os_awareness>.load-parameters`](#)

`<os_awareness>.wait-for-activated`

Synopsis

`<os_awareness>.wait-for-activated [-always] [-immediately] node`

Description

This command will wait until a processor known by the tracker system is active on a node matching the node path pattern.

The *node* parameter is a *node path pattern* that identifies the software node that should become active. See the *Analyzer User's Guide* for more information about node path patterns. The node path pattern will be evaluated every time a node becomes active in order to check if the wait condition is fulfilled. This means that a node that did not exist when this command was issued might cause this command to return. There is also a performance penalty involved. If only one node is of interest, a node id can be given (as an integer value) instead.

If the *-immediately* flag is specified and a processor is active on any of the nodes matching the node path pattern, this command will return directly, otherwise the command will wait until the set of active processors on the matching nodes goes from being empty to being non-empty.

The *-always* flag is internal and should not be used.

See Also

[`<os_awareness>.wait-for-deactivated`](#), [`<os_awareness>.break-enter`](#), [`<os_awareness>.break-exit`](#)

`<os_awareness>.wait-for-active — deprecated`

Synopsis

`<os_awareness>.wait-for-active [-always] node`

Description

This command is deprecated; use [`wait-for-activated`](#) instead.

`<os_awareness>.wait-for-deactivated`

Synopsis

`<os_awareness>.wait-for-deactivated [-always] [-immediately] node`

Description

This command will wait until no processor known by the tracker system is active on any node matching the node path pattern.

The *node* parameter is a *node path pattern* that identifies the software node that should become inactive. See the *Analyzer User's Guide* for more information about node path patterns. The node path pattern will be evaluated every time a node becomes inactive in order to check if the wait condition is fulfilled. This means that a node that did not exist when this command was issued might cause this command to continue to wait. There is also a performance penalty involved. If only one node is of interest, a node id can be given (as an integer value) instead.

If the *-immediately* flag is specified and no processor is active on any of the nodes matching the node path pattern, this command will return directly, otherwise the command will wait until the set of active processors on the matching nodes goes from being non-empty to being empty.

The *-always* flag is internal and should not be used.

See Also

[`<os_awareness>.wait-for-activated`](#), [`<os_awareness>.break-enter`](#), [`<os_awareness>.break-exit`](#)

`<os_awareness>.wait-for-inactive — deprecated`

Synopsis

`<os_awareness>.wait-for-inactive [-always] node`

Description

This command is deprecated; use [`wait-for-deactivated`](#) instead.

`<os_awareness>.wr-hypervisor-detect-settings`

Synopsis

`<os_awareness>.wr-hypervisor-detect-settings symbol-file [param-file] ["version-string"] [-overwrite]`

Description

This command generates a parameter file for a WR hypervisor configuration. It takes the path to a file containing symbols for the WR hypervisor system, given by the *file* argument. The resulting parameter file will be written to *output* or 'wrhyper.params' if *output* is left out. The *overwrite* argument can be used in order to allow overwrite of an old file with new parameters.

pc-dual-serial-ports

Provided By

[isa-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pc-dual-serial-ports” component represents two PC compatible serial ports.

Connectors

Name	Type	Direction
isa-bus	isa-bus	up
com[1-2]	serial	down

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<pc-dual-serial-ports>.info

Synopsis

<pc-dual-serial-ports>.info

Description

Print detailed information about the configuration of the device.

<pc-dual-serial-ports>.status

Synopsis

<pc-dual-serial-ports>.status

Description

Print detailed information about the current status of the device.

pc-floppy-controller

Provided By

[isa-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pc-floppy-controller” component represents a legacy pc floppy controller with two attached drives.

Connectors

Name	Type	Direction
isa-bus	isa-bus	up

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<pc-floppy-controller>.info

Synopsis

<pc-floppy-controller>.info

Description

Print detailed information about the configuration of the device.

<pc-floppy-controller>.status

Synopsis

<pc-floppy-controller>.status

Description

Print detailed information about the current status of the device.

pc-quad-serial-ports

Provided By
[isa-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pc-quad-serial-ports” component represents four PC compatible serial ports.

Connectors

Name	Type	Direction
isa-bus	isa-bus	up
com[1-4]	serial	down

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<pc-quad-serial-ports>.info

Synopsis

<pc-quad-serial-ports>.info

Description

Print detailed information about the configuration of the device.

<pc-quad-serial-ports>.status

Synopsis

<pc-quad-serial-ports>.status

Description

Print detailed information about the current status of the device.

pc-single-parallel-port

Provided By

[isa-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pc-single-parallel-port” component represents a PC compatible parallel port.

Connectors

Name	Type	Direction
isa-bus	isa-bus	up

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<pc-single-parallel-port>.info

Synopsis

<pc-single-parallel-port>.info

Description

Print detailed information about the configuration of the device.

<pc-single-parallel-port>.status

Synopsis

<pc-single-parallel-port>.status

Description

Print detailed information about the current status of the device.

pci-accel-vga

Provided By
[pci-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pci-accel-vga” component represents a PCI based VGA compatible graphics adapter.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
console	graphics-console	down

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<pci-accel-vga>.info

Synopsis

<pci-accel-vga>.info

Description

Print detailed information about the configuration of the device.

<pci-accel-vga>.status

Synopsis

<pci-accel-vga>.status

Description

Print detailed information about the current status of the device.

pci-am79c973

Provided By
[pci-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pci-am79c973” component represents a AM79C973 PCI based Ethernet adapter.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
ethernet	ethernet-link	down

Attributes

bios

Optional attribute; **read/write** access; type: `string` or `nil`. The x86 BIOS file to use.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

mac_address

Required attribute; **read/write** access; type: `string`. The MAC address of the Ethernet adapter.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<pci-am79c973>.info

Synopsis

<pci-am79c973>.info

Description

Print detailed information about the configuration of the device.

<pci-am79c973>.status

Synopsis

<pci-am79c973>.status

Description

Print detailed information about the current status of the device.

pci-bcm5703c

Provided By

[pci-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pci-bcm5703c” component represents a Broadcom 5703C PCI based gigabit Ethernet adapter.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
ethernet	ethernet-link	down

Attributes

bios

Optional attribute; **read/write** access; type: `string` or `nil`. The x86 BIOS file to use.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

mac_address

Required attribute; **read/write** access; type: `string`. The MAC address of the Ethernet adapter.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the `set-component-prefix` command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<pci-bcm5703c>.info

Synopsis

<pci-bcm5703c>.info

Description

Print detailed information about the configuration of the device.

<pci-bcm5703c>.status

Synopsis

<pci-bcm5703c>.status

Description

Print detailed information about the current status of the device.

pci-bcm5704c

Provided By
[pci-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pci-bcm5704c” component represents a Broadcom 5704C PCI based dual-port gigabit Ethernet adapter.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
ethernet[0-1]	ethernet-link	down

Attributes

bios

Optional attribute; **read/write** access; type: `string` or `nil`. The x86 BIOS file to use.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

mac_address0

Required attribute; **read/write** access; type: `string`. The MAC address of the first Ethernet adapter.

mac_address1

Required attribute; **read/write** access; type: `string`. The MAC address of the second Ethernet adapter.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the `set-component-prefix` command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

`<pci-bcm5704c>.info`

Synopsis

`<pci-bcm5704c>.info`

Description

Print detailed information about the configuration of the device.

`<pci-bcm5704c>.status`

Synopsis

`<pci-bcm5704c>.status`

Description

Print detailed information about the current status of the device.

pci-dec21041

Provided By

[pci-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pci-dec21041” component represents an Intel DEC21041 PCI based fast Ethernet adapter.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
ethernet	ethernet-link	down

Attributes

bios

Optional attribute; **read/write** access; type: `string` or `nil`. The x86 BIOS file to use.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

mac_address

Required attribute; **read/write** access; type: `string`. The MAC address of the Ethernet adapter.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<pci-dec21041>.info

Synopsis

<pci-dec21041>.info

Description

Print detailed information about the configuration of the device.

<pci-dec21041>.status

Synopsis

<pci-dec21041>.status

Description

Print detailed information about the current status of the device.

pci-dec21140a

Provided By

[pci-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pci-dec21140a” component represents an Intel DEC21140A PCI based fast Ethernet adapter.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
ethernet	ethernet-link	down

Attributes

bios

Optional attribute; **read/write** access; type: `string` or `nil`. The x86 BIOS file to use.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

mac_address

Required attribute; **read/write** access; type: `string`. The MAC address of the Ethernet adapter.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<pci-dec21140a>.info

Synopsis

<pci-dec21140a>.info

Description

Print detailed information about the configuration of the device.

<pci-dec21140a>.status

Synopsis

<pci-dec21140a>.status

Description

Print detailed information about the current status of the device.

pci-dec21140a-dml

Provided By

[pci-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pci-dec21140a-dml” component represents a DEC21140A PCI based fast Ethernet adapter (modeled in DML).

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
ethernet	ethernet-link	down

Attributes

bios

Optional attribute; **read/write** access; type: `string` or `nil`. The x86 BIOS file to use.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

mac_address

Required attribute; **read/write** access; type: `string`. The MAC address of the Ethernet adapter.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<pci-dec21140a-dml>.info

Synopsis

<pci-dec21140a-dml>.info

Description

Print detailed information about the configuration of the device.

<pci-dec21140a-dml>.status

Synopsis

<pci-dec21140a-dml>.status

Description

Print detailed information about the current status of the device.

pci-dec21143

Provided By

[pci-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pci-dec21143” component represents an Intel DEC21143 PCI based fast Ethernet adapter.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
ethernet	ethernet-link	down

Attributes

bios

Optional attribute; **read/write** access; type: `string` or `nil`. The x86 BIOS file to use.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

mac_address

Required attribute; **read/write** access; type: `string`. The MAC address of the Ethernet adapter.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the `set-component-prefix` command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<pci-dec21143>.info

Synopsis

<pci-dec21143>.info

Description

Print detailed information about the configuration of the device.

<pci-dec21143>.status

Synopsis

<pci-dec21143>.status

Description

Print detailed information about the current status of the device.

pci-i21152

Provided By
[pci-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pci-i21152” component represents an Intel® 21152 Transparent PCI-to-PCI Bridge.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
pci-slot[0-23]	pci-bus	any

Attributes

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<pci-i21152>.info

Synopsis

<pci-i21152>.info

Description

Print detailed information about the configuration of the device.

<pci-i21152>.status

Synopsis

<pci-i21152>.status

Description

Print detailed information about the current status of the device.

pci-i82543gc

Provided By

[pci-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pci-i82543gc” component represents the PCI-based Intel® 82543 Gigabit Ethernet Controller.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
ethernet	ethernet-link	down

Attributes

bios

Optional attribute; **read/write** access; type: `string` or `nil`. The x86 BIOS file to use.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

mac_address

Required attribute; **read/write** access; type: `string`. The MAC address of the Ethernet adapter.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<pci-i82543gc>.info

Synopsis

<pci-i82543gc>.info

Description

Print detailed information about the configuration of the device.

<pci-i82543gc>.status

Synopsis

<pci-i82543gc>.status

Description

Print detailed information about the current status of the device.

pci-i82546bg

Provided By

[pci-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pci-i82546bg” component represents an Intel® 82546 Gigabit Ethernet Controller.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
ethernet[0-1]	ethernet-link	down

Attributes

bios

Optional attribute; **read/write** access; type: `string` or `nil`. The x86 BIOS file to use.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

mac_address

Required attribute; **read/write** access; type: `string`. The MAC address of the first Ethernet adapter. The last bit is toggled to get the address for the second interface.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

`<pci-i82546bg>.info`

Synopsis

`<pci-i82546bg>.info`

Description

Print detailed information about the configuration of the device.

`<pci-i82546bg>.status`

Synopsis

`<pci-i82546bg>.status`

Description

Print detailed information about the current status of the device.

pci-i82559

Provided By
[pci-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

An Ethernet device with Intel® 82559 Fast Ethernet Controller.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
ethernet	ethernet-link	down

Attributes

bios

Optional attribute; **read/write** access; type: `string` or `nil`. The x86 BIOS file to use.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

mac_address

Required attribute; **read/write** access; type: `string`. The MAC address of this NIC

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<pci-i82559>.info

Synopsis

<pci-i82559>.info

Description

Print detailed information about the configuration of the device.

<pci-i82559>.status

Synopsis

<pci-i82559>.status

Description

Print detailed information about the current status of the device.

pci-ispl040

Provided By

[pci-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pci-ispl040” component represents an ISP1040 PCI based SCSI controller.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
scsi-bus	scsi-bus	down

Attributes

bios

Optional attribute; **read/write** access; type: `string` or `nil`. The x86 SCSI BIOS file to use.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

scsi_id

Optional attribute; **read/write** access; type: `integer`. The ID on the SCSI bus.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<pci-ispl040>.info

Synopsis

<pci-ispl040>.info

Description

Print detailed information about the configuration of the device.

<pci-ispl040>.status

Synopsis

<pci-ispl040>.status

Description

Print detailed information about the current status of the device.

pci-ispl2200

Provided By
[pci-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pci-ispl2200” component represents an ISP2200 PCI based Fibre Channel SCSI controller.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
fc-loop	simple-fc-loop	down

Attributes

bios

Optional attribute; **read/write** access; type: `string` or `nil`. The x86 SCSI BIOS file to use.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

loop_id

Required attribute; **read/write** access; type: `integer`. The FC loop ID of the ISP2200 Fibre Channel controller.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<pci-ispl2200>.info

Synopsis

<pci-ispl2200>.info

Description

Print detailed information about the configuration of the device.

<pci-ispl2200>.status

Synopsis

<pci-ispl2200>.status

Description

Print detailed information about the current status of the device.

pci-pd6729

Provided By
[pci-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pci-pd6729” component represents a Cirrus Logic PD6729 PCI-to-PCMCIA (PC-Card) Controller with two slots.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
pcmcia[0-1]	pcmcia-slot	down

Attributes

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<pci-pd6729>.info

Synopsis

<pci-pd6729>.info

Description

Print detailed information about the configuration of the device.

<pci-pd6729>.status

Synopsis

<pci-pd6729>.status

Description

Print detailed information about the current status of the device.

pci-pmc1553

Provided By

[mil-std-1553-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pci-pmc1553” component represents a PMC-1553PCI based MIL-STD-1553 Bus Controller.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
link-a	ms1553-link	down
link-b	ms1553-link	down

Attributes

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<pci-pmc1553>.info

Synopsis

<pci-pmc1553>.info

Description

Print detailed information about the configuration of the device.

<pci-pmc1553>.status

Synopsis

<pci-pmc1553>.status

Description

Print detailed information about the current status of the device.

pci-sil680a

Provided By

[pci-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

SIL680a IDE controller

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
ide0-slot[0-1]	ide-slot	down
ide1-slot[0-1]	ide-slot	down

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<pci-sil680a>.info

Synopsis

<pci-sil680a>.info

Description

Print detailed information about the configuration of the device.

<pci-sil680a>.status

Synopsis

<pci-sil680a>.status

Description

Print detailed information about the current status of the device.

pci-sym53c810

Provided By

[pci-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pci-sym53C810” component represents a SYM53C810PCI based SCSI controller.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
scsi-bus	scsi-bus	down

Attributes

bios

Optional attribute; **read/write** access; type: `string` or `nil`. The x86 SCSI BIOS file to use.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<pci-sym53c810>.info

Synopsis

<pci-sym53c810>.info

Description

Print detailed information about the configuration of the device.

<pci-sym53c810>.status

Synopsis

<pci-sym53c810>.status

Description

Print detailed information about the current status of the device.

pci-sym53c875

Provided By

[pci-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pci-sym53C875” component represents a SYM53C875PCI based SCSI controller.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
scsi-bus	scsi-bus	down

Attributes

bios

Optional attribute; **read/write** access; type: `string` or `nil`. The x86 SCSI BIOS file to use.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<pci-sym53c875>.info

Synopsis

<pci-sym53c875>.info

Description

Print detailed information about the configuration of the device.

<pci-sym53c875>.status

Synopsis

<pci-sym53c875>.status

Description

Print detailed information about the current status of the device.

pci-sym53c876

Provided By
[pci-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pci-sym53C876” component represents a SYM53C876PCI based dual-port SCSI controller.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
scsi-bus[0-1]	scsi-bus	down

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<pci-sym53c876>.info

Synopsis

<pci-sym53c876>.info

Description

Print detailed information about the configuration of the device.

<pci-sym53c876>.status

Synopsis

<pci-sym53c876>.status

Description

Print detailed information about the current status of the device.

pci-tsb12lv26

Provided By

[firewire-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

A tsb12lv26 component containing a TSB12LV26 PCI device.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
firewire-bus	firewire	down

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<pci-tsb12lv26>.info

Synopsis

<pci-tsb12lv26>.info

Description

Print detailed information about the configuration of the device.

<pci-tsb12lv26>.status

Synopsis

<pci-tsb12lv26>.status

Description

Print detailed information about the current status of the device.

pci-vga

Provided By

[pci-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “pci-vga” component represents a PCI based VGA compatible graphics adapter.

Connectors

Name	Type	Direction
pci-bus	pci-bus	up
console	graphics-console	down

Attributes

bios

Optional attribute; **read/write** access; type: `string` or `nil`. The VGA BIOS file to use.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Pseudo class attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<pci-vga>.info

Synopsis

<pci-vga>.info

Description

Print detailed information about the configuration of the device.

<pci-vga>.status

Synopsis

<pci-vga>.status

Description

Print detailed information about the current status of the device.

phy-mii-transceiver

Provided By
[phy-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

Obsolete PHY component representing a generic MII transceiver, based on the **mii-transceiver** object. This component has been replaced by the **phy_comp** component, which is based on the new **generic_eth_phy** model

Connectors

Name	Type	Direction
mac	phy	up
eth	ethernet-link	down

Attributes

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

mii_address

Optional attribute; **read/write** access; type: `integer`. PHY address on MII bus

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

phy_id

Optional attribute; **read/write** access; type: `integer`. PHY ID (i.e. vendor)

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<**phy-mii-transceiver**>.info

Synopsis

<**phy-mii-transceiver**>.info

Description

Print detailed information about the configuration of the device.

<**phy-mii-transceiver**>.status

Synopsis

<**phy-mii-transceiver**>.status

Description

Print detailed information about the current status of the device.

phy_comp

Provided By

[phy-comp](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

Component representing a generic IEEE 802.3 PHY

Connectors

Name	Type	Direction
mac	phy	up
eth	ethernet-link	down

Command List

Commands

[info](#) print information about the device

[status](#) print status of the device

Command Descriptions

<phy_comp>.info

Synopsis

[<phy_comp>.info](#)

Description

Print detailed information about the configuration of the device.

<phy_comp>.status

Synopsis

[<phy_comp>.status](#)

Description

Print detailed information about the current status of the device.

ps2-keyboard-mouse

Provided By

[isa-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “ps2-keyboard-mouse” component represents the PS/2 8042 keyboard controller with a connected 105 key keyboard and three button mouse.

Connectors

Name	Type	Direction
isa-bus	isa-bus	up
reset	x86-reset-bus	up
kbd-console	keyboard	down
mse-console	mouse	down

Attributes

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<ps2-keyboard-mouse>.info

Synopsis

<ps2-keyboard-mouse>.info

Description

Print detailed information about the configuration of the device.

<ps2-keyboard-mouse>.status

Synopsis

<ps2-keyboard-mouse>.status

Description

Print detailed information about the current status of the device.

rapidio_link

Provided By
[rapidio-link](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description
RapidIO link connecting two RapidIO devices

Connectors

Name	Type	Direction
device[0-1]	rapidio	any

Attributes

global_id

Optional attribute; **read/write** access; type: `string` or `nil`. Global identifier for use in distributed simulation or `NIL` if the link is not distributed.

goal_latency

Optional attribute; **read/write** access; type: `float`. Goal latency in seconds for this link. See also the `set-min-latency` command.

Command List

Commands

[**info**](#) print information about the device

[**status**](#) print status of the device

Command Descriptions

`<rapidio_link>.info`

Synopsis

`<rapidio_link>.info`

Description

Print detailed information about the configuration of the device.

`<rapidio_link>.status`

Synopsis

`<rapidio_link>.status`

Description

Print detailed information about the current status of the device.

real-network-bridge

Provided By
[real-network](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “real-network-bridge” component represents a bridged connection to a real network

Connectors

Name	Type	Direction
link	ethernet-link	down

Attributes

access

Required attribute; **read/write** access; type: `string`. access type (“raw” or “tap”)

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

no_mac_translate

Required attribute; **read/write** access; type: `boolean`. whether mac address translation should be skipped

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

disconnect-real-network	disconnect from the real network
finish-connection	Finalize a real network connection
info	print information about the device
status	print status of the device

Command Descriptions

`<real-network-bridge>.disconnect-real-network`

Synopsis

`<real-network-bridge>.disconnect-real-network`

Description

Disconnect the real network connection from a simulated Ethernet link.

`<real-network-bridge>.finish-connection`

Synopsis

`<real-network-bridge>.finish-connection`

Description

Finish the connection to the real network once the TAP interface has been assigned an IP address and network mask ethat the real-network object has to read. This command usually isn't needed since Simics will retry getting the interface configuration when it receives a packet from or to the real network.

`<real-network-bridge>.info`

Synopsis

`<real-network-bridge>.info`

Description

Print detailed information about the configuration of the device.

`<real-network-bridge>.status`

Synopsis

`<real-network-bridge>.status`

Description

Print detailed information about the current status of the device.

real-network-host

Provided By
[real-network](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “real-network-host” component represents a host-based connection to a real network

Connectors

Name	Type	Direction
link	ethernet-link	down

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

interface

Required attribute; **read/write** access; type: string. Interface to connect to

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

disconnect-real-network	disconnect from the real network
finish-connection	Finalize a real network connection
info	print information about the device
status	print status of the device

Command Descriptions

<real-network-host>.disconnect-real-network

Synopsis

<real-network-host>.disconnect-real-network

Description

Disconnect the real network connection from a simulated Ethernet link.

<real-network-host>.finish-connection

Synopsis

<real-network-host>.finish-connection

Description

Finish the connection to the real network once the TAP interface has been assigned an IP address and network mask ethat the real-network object has to read. This command usually isn't needed since Simics will retry getting the interface configuration when it receives a packet from or to the real network.

<real-network-host>.info

Synopsis

<real-network-host>.info

Description

Print detailed information about the configuration of the device.

<real-network-host>.status

Synopsis

<real-network-host>.status

Description

Print detailed information about the current status of the device.

real-network-router

Provided By
[real-network](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “real-network-router” component represents a router-based connection to a real network

Connectors

Name	Type	Direction
link	ethernet-link	down

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

ip

Required attribute; **read/write** access; type: string. IP

netmask

Required attribute; **read/write** access; type: string. netmask

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

disconnect-real-network	disconnect from the real network
info	print information about the device
status	print status of the device

Command Descriptions

`<real-network-router>.disconnect-real-network`

Synopsis

`<real-network-router>.disconnect-real-network`

Description

Disconnect the real network connection from a simulated Ethernet link.

`<real-network-router>.info`

Synopsis

`<real-network-router>.info`

Description

Print detailed information about the configuration of the device.

`<real-network-router>.status`

Synopsis

`<real-network-router>.status`

Description

Print detailed information about the current status of the device.

sample-gcache

Provided By
[timing-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description
A pre-configured combined L1 instruction and data cache

Connectors

Name	Type	Direction
prev-level	cache	up
next-level	cache	down

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<sample-gcache>.info

Synopsis

<sample-gcache>.info

Description

Print detailed information about the configuration of the device.

<sample-gcache>.status

Synopsis

<sample-gcache>.status

Description

Print detailed information about the current status of the device.

sdram-memory-module

Provided By
[memory-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “sdram-memory-module” component represents a SDRAM memory module.

Connectors

Name	Type	Direction
mem_bus	mem-bus	up

Attributes

banks

Optional attribute; **read/write** access; type: `integer`. Number of banks.

cas_latency

Optional attribute; **read/write** access; type: `integer`. CAS-latency; each set bit corresponds to a latency the memory can handle

columns

Optional attribute; **read/write** access; type: `integer`. Number of columns.

ecc_width

Optional attribute; **read/write** access; type: `integer`. The error correction width.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

module_data_width

Optional attribute; **read/write** access; type: `integer`. The module SDRAM width.

module_type

Optional attribute; **read/write** access; type: `string`. Type of memory.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

primary_width

Optional attribute; **read/write** access; type: `integer`. Primary SDRAM width.

rank_density

Optional attribute; **read/write** access; type: `integer`. The rank density.

ranks

Optional attribute; **read/write** access; type: `integer`. Number of ranks (logical banks).

rows

Optional attribute; **read/write** access; type: `integer`. Number of rows.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<s dram-memory-module>.info

Synopsis

<s dram-memory-module>.info

Description

Print detailed information about the configuration of the device.

<s dram-memory-module>.status

Synopsis

<sram-memory-module>.status

Description

Print detailed information about the current status of the device.

ser_link

Provided By
[ser-link](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

Serial link connecting two serial devices

Connectors

Name	Type	Direction
device[0-1]	serial	any

Attributes

global_id

Optional attribute; **read/write** access; type: `string` or `nil`. Global identifier for use in distributed simulation or `NIL` if the link is not distributed.

goal_latency

Optional attribute; **read/write** access; type: `float`. Goal latency in seconds for this link. See also the `set-min-latency` command.

Command List

Commands

[**info**](#) print information about the device

[**status**](#) print status of the device

Command Descriptions

<ser_link>.info

Synopsis

`<ser_link>.info`

Description

Print detailed information about the configuration of the device.

<ser_link>.status

Synopsis

`<ser_link>.status`

Description

Print detailed information about the current status of the device.

signal_link

Provided By

[signal-link](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “signal_link” component represents a unidirectional signal that can be either high or low. It can be used to model electrical wires or more abstract binary signals.

Connectors

Name	Type	Direction
receiver0	signal-link-receiver	any
sender0	signal-link-sender	any

Attributes

global_id

Optional attribute; **read/write** access; type: `string` or `nil`. Global identifier for use in distributed simulation or `NIL` if the link is not distributed.

goal_latency

Optional attribute; **read/write** access; type: `float`. Goal latency in seconds for this link. See also the `set-min-latency` command.

Command List

Commands

[**info**](#) print information about the device

[**status**](#) print status of the device

Command Descriptions

<signal_link>.info

Synopsis

`<signal_link>.info`

Description

Print detailed information about the configuration of the device.

<signal_link>.status

Synopsis

<signal_link>.status

Description

Print detailed information about the current status of the device.

simple-fc-disk

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “simple-fc-disk” component represents a SCSI-2 disk for use with Fibre Channel SCSI controllers using the simplified FC protocol in Simics.

Connectors

Name	Type	Direction
fc-loop	simple-fc-loop	up

Attributes

file

Optional attribute; **read/write** access; type: `string` or `nil`. File with disk contents for the full disk Either a raw file or a CRAFF file.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

loop_id

Required attribute; **read/write** access; type: `integer`. The loop ID for the FC disk.

node_name

Required attribute; **read/write** access; type: `integer`. The node name for the FC disk.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the `set-component-prefix` command before the component is created.

port_name

Required attribute; **read/write** access; type: `integer`. The port name for the FC disk.

size

Required attribute; **read/write** access; type: `integer`. The size of the FC disk in bytes.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

`<simple-fc-disk>.info`

Synopsis

`<simple-fc-disk>.info`

Description

Print detailed information about the configuration of the device.

`<simple-fc-disk>.status`

Synopsis

`<simple-fc-disk>.status`

Description

Print detailed information about the current status of the device.

simple_memory_module

Provided By

[memory-comp](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

A simple memory module.

Connectors

Name	Type	Direction
mem_bus	mem-bus	up

Command List

Commands

[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

<simple_memory_module>.info

Synopsis

[**<simple_memory_module>.info**](#)

Description

Print detailed information about the configuration of the device.

<simple_memory_module>.status

Synopsis

[**<simple_memory_module>.status**](#)

Description

Print detailed information about the current status of the device.

sio-w83627hf

Provided By
[isa-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description
Winbond W83627HF I/O.

Connectors

Name	Type	Direction
i2c-bus	i2c-bus	up
isa-bus	isa-bus	up
reset	x86-reset-bus	up
com[1-2]	serial	down
kbd-console	keyboard	down
mse-console	mouse	down

Attributes

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<sio-w83627hf>.info

Synopsis

<sio-w83627hf>.info

Description

Print detailed information about the configuration of the device.

<sio-w83627hf>.status

Synopsis

<sio-w83627hf>.status

Description

Print detailed information about the current status of the device.

std-etg

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “std-etg” component represents an Ethernet traffic generator.

Connectors

Name	Type	Direction
ethernet	ethernet-link	down

Attributes

dst_ip

Required attribute; **read/write** access; type: `string`. Destination IP address for generated traffic.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

ip

Required attribute; **read/write** access; type: `string`. IP address of the traffic generator.

netmask

Required attribute; **read/write** access; type: `string`. IP netmask of the traffic generator.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the `set-component-prefix` command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

change-reference-clock	change the reference clock of an etg component
info	print information about the device
status	print status of the device

Command Descriptions

`<std-etg>.change-reference-clock`

Synopsis

`<std-etg>.change-reference-clock arg1`

Description

Change the reference clock of an etg component, if the automatically chosen clock was not optimal for the current configuration.

`<std-etg>.info`

Synopsis

`<std-etg>.info`

Description

Print detailed information about the configuration of the device.

`<std-etg>.status`

Synopsis

`<std-etg>.status`

Description

Print detailed information about the current status of the device.

std-ethernet-link

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The obsolete **std-ethernet-link** component represents an Ethernet link model based on the old-style link **ethernet-link**. It is provided for compatibility with older checkpoints and setups. New configurations should be based on the **ethernet_switch**, **ethernet_hub** or **ethernet_cable** components instead.

Connectors

Name	Type	Direction
device	ethernet-link	any

Attributes

distributed

Optional attribute; **read/write** access; type: `boolean`. Connect this link to the Central client, so it can be distributed.

frame_echo

Optional attribute; **read/write** access; type: `integer`. Set this attribute to echo frames back to the sender. Default is not to echo frames.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

latency

Optional attribute; **read/write** access; type: `integer`. The latency in nanoseconds for communication over the link.

link_name

Optional attribute; **read/write** access; type: `string` or `nil`. The name to use for the object. An error will be raised at instantiation time if the link cannot be given this name.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

`<std-ethernet-link>.info`

Synopsis

`<std-ethernet-link>.info`

Description

Print detailed information about the configuration of the device.

`<std-ethernet-link>.status`

Synopsis

`<std-ethernet-link>.status`

Description

Print detailed information about the current status of the device.

std-firewire-bus

Provided By
[firewire-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description
A firewire bus

Connectors

Name	Type	Direction
device	firewire	any

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<std-firewire-bus>.info

Synopsis

<std-firewire-bus>.info

Description

Print detailed information about the configuration of the device.

<std-firewire-bus>.status

Synopsis

<std-firewire-bus>.status

Description

Print detailed information about the current status of the device.

std-firewire-sample-device

Provided By
[firewire-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description
A firewire sample device

Connectors

Name	Type	Direction
firewire-bus	firewire	up

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<std-firewire-sample-device>.info

Synopsis

<std-firewire-sample-device>.info

Description

Print detailed information about the configuration of the device.

<std-firewire-sample-device>.status

Synopsis

<std-firewire-sample-device>.status

Description

Print detailed information about the current status of the device.

std-generic-link

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The obsolete **std-generic-link** component represents a “generic” link model that can transmit messages from one point to another. It is provided for compatibility with older checkpoints and setups. New configurations should either use the simpler **datagram-link** component, for broadcast or point-to-point usage, or use the Simics Link Library to create a custom link, for more complex usage. Refer to the *Link Library Programming Guide* for more information.

Connectors

Name	Type	Direction
device	generic-link	any

Attributes

distributed

Optional attribute; **read/write** access; type: `boolean`. Connect this link to the Central client, so it can be distributed.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

latency

Optional attribute; **read/write** access; type: `integer`. The latency in nanoseconds for communication over the link.

link_name

Optional attribute; **read/write** access; type: `string` or `nil`. The name to use for the object. An error will be raised at instantiation time if the link cannot be given this name.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

`<std-generic-link>.info`

Synopsis

`<std-generic-link>.info`

Description

Print detailed information about the configuration of the device.

`<std-generic-link>.status`

Synopsis

`<std-generic-link>.status`

Description

Print detailed information about the current status of the device.

std-generic-link-sample

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The obsolete **std-generic-link-sample** component represents a generic link sample device that can connect to a **std-generic-link** link. New configurations should not be based on generic-links: they should either use the simpler **datagram-link** component, for broadcast or point-to-point usage, or use the Simics Link Library to create a custom link, for more complex usage. Refer to the *Link Library Programming Guide* for more information.

Connectors

Name	Type	Direction
link	generic-link	down

Attributes

address

Required attribute; **read/write** access; type: `integer`. The address of the sample device.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<std-generic-link-sample>.info

Synopsis

<std-generic-link-sample>.info

Description

Print detailed information about the configuration of the device.

<std-generic-link-sample>.status

Synopsis

<std-generic-link-sample>.status

Description

Print detailed information about the current status of the device.

std-graphics-console

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “std-graphics-console” component represents a graphical console for displaying output from a simulated graphics adapters and getting input for mouse and keyboard devices.

Connectors

Name	Type	Direction
abs-mouse	abs-mouse	up
device	graphics-console	up
keyboard	keyboard	up
mouse	mouse	up
clipboard	clipboard	down

Attributes

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

title

Optional attribute; **read/write** access; type: `string`. The Window title.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

window

Optional attribute; **read/write** access; type: `boolean`. Try to open window if TRUE (default). FALSE disables the window.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

<code>info</code>	print information about the device
<code>status</code>	print status of the device
<code>switch-to-text-console</code>	replace the console with a text console

Command Descriptions

`<std-graphics-console>.info`

Synopsis

`<std-graphics-console>.info`

Description

Print detailed information about the configuration of the device.

`<std-graphics-console>.status`

Synopsis

`<std-graphics-console>.status`

Description

Print detailed information about the current status of the device.

`<std-graphics-console>.switch-to-text-console`

Synopsis

`<std-graphics-console>.switch-to-text-console`

Description

Replace the std-graphics-console component with a std-text-graphics-console component.

See Also

[new-std-text-graphics-console](#)

std-host-serial-console

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “std-host-serial console” component represents a serial console accessible from the host as a virtual serial port. On Windows a COM port is used and on Unix a pty (pseudo-terminal). The name of an existing port can be supplied when the console is created. Then this port will be used, and no pseudo port is opened.

Connectors

Name	Type	Direction
serial	serial	up

Attributes

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info	print information about the device
status	print status of the device
switch-to-telnet-console	replace the console with a telnet console
switch-to-text-console	replace the console with a text console

Command Descriptions

<std-host-serial-console>.info

Synopsis

<std-host-serial-console>.info

Description

Print detailed information about the configuration of the device.

<std-host-serial-console>.status

Synopsis

<std-host-serial-console>.status

Description

Print detailed information about the current status of the device.

<std-host-serial-console>.switch-to-telnet-console

Synopsis

<std-host-serial-console>.switch-to-telnet-console [port]

Description

Replace the %s component with a std-text-console component. The argument *port* is the port number that a telnet client can connect to. If no argument is given, a free port on the host is used. This port number is returned by the command.

See Also

[new-std-telnet-console](#)

<std-host-serial-console>.switch-to-text-console

Synopsis

<std-host-serial-console>.switch-to-text-console

Description

Replace the std-host-serial-console component with a std-text-console component.

See Also

[new-std-telnet-console](#)

std-ide-cdrom

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “std-ide-cdrom” component represents an IDE ATAPI CD-ROM.

Connectors

Name	Type	Direction
ide-slot	ide-slot	up

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<std-ide-cdrom>.info

Synopsis

<std-ide-cdrom>.info

Description

Print detailed information about the configuration of the device.

<std-ide-cdrom>.status

Synopsis

<std-ide-cdrom>.status

Description

Print detailed information about the current status of the device.

std-ide-disk

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “std-ide-disk” component represents an IDE disk.

Connectors

Name	Type	Direction
ide-slot	ide-slot	up

Attributes

file

Optional attribute; **read/write** access; type: `string` or `nil`. File with disk contents for the full disk. Either a raw file or a CRAFF file.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

size

Required attribute; **read/write** access; type: `integer`. The size of the IDE disk in bytes.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<std-ide-disk>.info

Synopsis

<std-ide-disk>.info

Description

Print detailed information about the configuration of the device.

<std-ide-disk>.status

Synopsis

<std-ide-disk>.status

Description

Print detailed information about the current status of the device.

std-ms1553-link

Provided By

[mil-std-1553-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “std-ms1553-link” component represents a MIL-STD-1553 bus/link.

Connectors

Name	Type	Direction
device	ms1553-link	any

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<std-ms1553-link>.info

Synopsis

<std-ms1553-link>.info

Description

Print detailed information about the configuration of the device.

<std-ms1553-link>.status

Synopsis

<std-ms1553-link>.status

Description

Print detailed information about the current status of the device.

std-pcmcia-flash-disk

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “std-pcmcia-flash-disk” component represents a PCMCIA Flash disk.

Connectors

Name	Type	Direction
pcmcia-slot	pcmcia-slot	up

Attributes

file

Optional attribute; **read/write** access; type: `string` or `nil`. File with disk contents for the full disk. Either a raw file or a CRAFF file.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

size

Required attribute; **read/write** access; type: `integer`. The size of the IDE disk in bytes.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<std-pcmcia-flash-disk>.info

Synopsis

<std-pcmcia-flash-disk>.info

Description

Print detailed information about the configuration of the device.

<std-pcmcia-flash-disk>.status

Synopsis

<std-pcmcia-flash-disk>.status

Description

Print detailed information about the current status of the device.

std-scsi-bus

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “std-scsi-bus” component represents a 16 slot SCSI bus.

Connectors

Name	Type	Direction
scsi-bus	scsi-bus	any

Attributes

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<std-scsi-bus>.info

Synopsis

<std-scsi-bus>.info

Description

Print detailed information about the configuration of the device.

<std-scsi-bus>.status

Synopsis

<std-scsi-bus>.status

Description

Print detailed information about the current status of the device.

std-scsi-cdrom

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “std-scsi-cdrom” component represents a SCSI-2 CD-ROM.

Connectors

Name	Type	Direction
scsi-bus	scsi-bus	up

Attributes

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

scsi_id

Required attribute; **read/write** access; type: `integer`. The ID on the SCSI bus.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<std-scsi-cdrom>.info

Synopsis

<std-scsi-cdrom>.info

Description

Print detailed information about the configuration of the device.

<std-scsi-cdrom>.status

Synopsis

<std-scsi-cdrom>.status

Description

Print detailed information about the current status of the device.

std-scsi-disk

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “std-scsi-disk” component represents a SCSI-2 disk.

Connectors

Name	Type	Direction
scsi-bus	scsi-bus	up

Attributes

file

Optional attribute; **read/write** access; type: `string` or `nil`. File with disk contents for the full disk. Either a raw file or a CRAFF file.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

scsi_id

Required attribute; **read/write** access; type: `integer`. The ID on the SCSI bus.

size

Required attribute; **read/write** access; type: `integer`. The size of the SCSI disk in bytes.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<std-scsi-disk>.info

Synopsis

<std-scsi-disk>.info

Description

Print detailed information about the configuration of the device.

<std-scsi-disk>.status

Synopsis

<std-scsi-disk>.status

Description

Print detailed information about the current status of the device.

std-serial-link

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The obsolete **std-serial-link** component represents an serial link model based on the old-style link **serial-link**. It is provided for compatibility with older checkpoints and setups. New configurations should be based on the **ser_link** component instead.

Connectors

Name	Type	Direction
serial[0-1]	serial	any

Attributes

distributed

Optional attribute; **read/write** access; type: `boolean`. Connect this link to the Central client, so it can be distributed.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

latency

Optional attribute; **read/write** access; type: `integer`. The latency in nanoseconds for communication over the link.

link_name

Optional attribute; **read/write** access; type: `string` or `nil`. The name to use for the object. An error will be raised at instantiation time if the link cannot be given this name.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

throttle

Optional attribute; **read/write** access; type: `integer`. The maximum number of bits per second that may be sent over the link in each direction. A value of zero disables throttling

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

`<std-serial-link>.info`

Synopsis

`<std-serial-link>.info`

Description

Print detailed information about the configuration of the device.

`<std-serial-link>.status`

Synopsis

`<std-serial-link>.status`

Description

Print detailed information about the current status of the device.

std-server-console

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description
Deprecated class - do not use

Connectors

Name	Type	Direction
serial	serial	up

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<std-server-console>.info

Synopsis

<std-server-console>.info

Description

Print detailed information about the configuration of the device.

<std-server-console>.status

Synopsis

<std-server-console>.status

Description

Print detailed information about the current status of the device.

std-service-node

Provided By
[service-node](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “std-service-node” component represents a network service node that can be connected to Ethernet links to provide services such as DNS, DHCP/BOOTP, RARP and TFTP. A service node component does not have any connectors by default. Instead, connectors have to be added using the `<std-service-node>.add-connector` command.

Attributes

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the `set-component-prefix` command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

add-connector	add a service-node connector
add-host	add host entry
arp	inspect and manipulate ARP table
change-reference-clock	change the reference clock of a service-node component
connect-to-link	connect a service-node component to a link
delete-host	delete host entry
dhcp-add-pool	add DHCP pool
dhcp-leases	show DHCP leases
disable-real-dns	disable real DNS
disable-service	disable network service
enable-ftp-alg	enable FTP ALG
enable-real-dns	enable real DNS
enable-service	enable network service
info	print information about the device
list-host-info	print host info database
route	show the routing table
route-add	add an entry to the routing table
set-tftp-directory	set TFTP directory
status	print status of the device
tcpip-info	show TCP/IP info

Command Descriptions

<std-service-node>.add-connector

Synopsis

<std-service-node>.add-connector "ip" ["netmask"]

Description

Adds a connector to the service-node with specified IP address and netmask. A connector must be created for the service-node before an Ethernet link can be connected to it. The *ip* argument is the IP address that the service node will use on the link, and optionally a prefix length on the form "1.2.3.4/24". The *netmask* argument can be used instead of a prefix length for IPv4 addresses for backwards compatibility. The default prefix length for IPv4 addresses is 24, and for IPv6 addresses 64.

The name of the new connector is returned.

<std-service-node>.add-host

Synopsis

<std-service-node>.add-host "ip" "name" ["domain"] ["mac"]

Description

Add a host entry to the DHCP and DNS server tables.

See Also

[**<service-node>.list-host-info**](#)

<std-service-node>.arp

Synopsis

```
<std-service-node>.arp [-d] ["delete-ip"]
```

Description

Prints the ARP table for the service node if called with no arguments. An ARP entry can be deleted using the -d flag and the IP address.

<std-service-node>.change-reference-clock

Synopsis

```
<std-service-node>.change-reference-clock arg1
```

Description

Change the reference clock of a service-node component, if the automatically chosen clock was not optimal for the current configuration.

<std-service-node>.connect-to-link

Synopsis

```
<std-service-node>.connect-to-link link "ip" ["netmask"]
```

Description

Connect the service-node to a link with specified IP address and netmask. The *ip* argument is the IP address that the service node will use on the link, and optionally a prefix length on the form "1.2.3.4/24". The *netmask* argument can be used instead of a prefix length for IPv4 addresses for backwards compatibility. The default prefix length for IPv4 addresses is 24, and for IPv6 addresses 64.

<std-service-node>.delete-host

Synopsis

```
<std-service-node>.delete-host ["ip"] ["name"] ["domain"] ["mac"]
```

Description

Delete a host entry from the DHCP and DNS server tables.

See Also

[`<service-node>.list-host-info`](#)

<std-service-node>.dhcp-add-pool

Synopsis

```
<std-service-node>.dhcp-add-pool pool-size "ip" ["name"] ["domain"]
```

Description

Add an IP address pool to the DHCP server. The *pool-size* parameter defines the number of available addresses in the pool, starting with address *ip*. The DNS server will map the addresses to a name that is the *name* parameter with a number appended, in the *domain* domain.

```
<std-service-node>.dhcp-leases
```

Synopsis

```
<std-service-node>.dhcp-leases
```

Description

Print the list of active DHCP leases.

```
<std-service-node>.disable-real-dns
```

Synopsis

```
<std-service-node>.disable-real-dns
```

Description

Disable forwarding of DNS queries for unknown hosts to the real DNS server.

```
<std-service-node>.disable-service
```

Synopsis

```
<std-service-node>.disable-service ["name"] [-all]
```

Description

Disable a named network service in the service-node, or all of the *-all* flag is used.

```
<std-service-node>.enable-ftp-alg
```

Synopsis

```
<std-service-node>.enable-ftp-alg
```

Description

Enable the FTP ALG. FTP ALG processing is needed to support port forwarded FTP from the outside network to simulated machines.

```
<std-service-node>.enable-real-dns
```

Synopsis

```
<std-service-node>.enable-real-dns
```

Description

Enable forwarding of DNS queries for unknown hosts to the real DNS server.

<std-service-node>.enable-service

Synopsis

```
<std-service-node>.enable-service ["name"] [-all]
```

Description

Enable a named network service in the service-node, or all of the *-all* flag is used.

<std-service-node>.info

Synopsis

```
<std-service-node>.info
```

Description

Print detailed information about the configuration of the device.

<std-service-node>.list-host-info

Synopsis

```
<std-service-node>.list-host-info
```

Description

Print the host information database, used by the DHCP and DNS server.

See Also

[`<service-node>.add-host`](#)

<std-service-node>.route

Synopsis

```
<std-service-node>.route [-port]
```

Description

Print the routing table.

By default, the name of the link on which traffic will be send is printed in the last column, but if the *-port* flag is use, the port device will be printed instead.

<std-service-node>.route-add

Synopsis

```
<std-service-node>.route-add "net" ["netmask"] ["gateway"] link
```

Description

Add to the routing table.

<std-service-node>.set-tftp-directory

Synopsis

```
<std-service-node>.set-tftp-directory (dir|-default)
```

Description

Set the directory used by the TFTP service for files read and written over TFTP. By default, the *SIM_lookup_file* is used for finding files to read, and the current directory is used for writing files.

```
<std-service-node>.status
```

Synopsis

```
<std-service-node>.status
```

Description

Print detailed information about the current status of the device.

```
<std-service-node>.tcpip-info
```

Synopsis

```
<std-service-node>.tcpip-info
```

Description

Print all TCP/IP connections.

std-super-io

Provided By
[isa-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “std-super-io” component represents a generic Super I/O device with legacy PC devices such as two serial ports, one PS/2 keyboard and mouse controller, one floppy device and a parallel port

Connectors

Name	Type	Direction
isa-bus	isa-bus	up
reset	x86-reset-bus	up
com[1-2]	serial	down
kbd-console	keyboard	down
mse-console	mouse	down

Attributes

add_par_port

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE to add a parallel port to the Super I/O device. Default is FALSE since the current implementation is a dummy device.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

`<std-super-io>.info`

Synopsis

`<std-super-io>.info`

Description

Print detailed information about the configuration of the device.

`<std-super-io>.status`

Synopsis

`<std-super-io>.status`

Description

Print detailed information about the current status of the device.

std-telnet-console

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “std-telnet-console” component represents a serial console accessible from the host using telnet.

Connectors

Name	Type	Direction
serial	serial	up

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info	print information about the device
status	print status of the device
switch-to-host-serial-console	replace the console with a host serial console)
switch-to-text-console	replace the console with a text console

Command Descriptions

`<std-telnet-console>.info`

Synopsis

`<std-telnet-console>.info`

Description

Print detailed information about the configuration of the device.

`<std-telnet-console>.status`

Synopsis

`<std-telnet-console>.status`

Description

Print detailed information about the current status of the device.

`<std-telnet-console>.switch-to-host-serial-console`

Synopsis

`<std-telnet-console>.switch-to-host-serial-console ["port"]`

Description

Replace the std-telnet-console component with a std-host-serial-console component. The argument *port* is an existing COM port on Windows and a TTY on Linux. If no argument is given, a new pseudo port is opened. The name of the host device is returned.

See Also

[new-std-host-serial-console](#)

`<std-telnet-console>.switch-to-text-console`

Synopsis

`<std-telnet-console>.switch-to-text-console`

Description

Replace the std-telnet-console component with a std-text-console component.

See Also

[new-std-telnet-console](#)

std-text-console

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “std-text-console” component represents a serial text console.

Connectors

Name	Type	Direction
serial	serial	up

Attributes

bg_color

Optional attribute; **read/write** access; type: `string`. The background color.

fg_color

Optional attribute; **read/write** access; type: `string`. The foreground color.

height

Optional attribute; **read/write** access; type: `integer`. The height of the console window.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

title

Optional attribute; **read/write** access; type: `string`. The Window title.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

width

Optional attribute; **read/write** access; type: `integer`. The width of the console window.

win32_font

Optional attribute; **read/write** access; type: `string`. Font to use in the console on Windows host.

window

Optional attribute; **read/write** access; type: `boolean`. Try to open window if TRUE (default). FALSE disables the window.

x11_font

Optional attribute; **read/write** access; type: `string`. Font to use in the console when using X11 (Linux/Solaris host).

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device

status print status of the device

switch-to-host-serial-console replace the console with a host serial console)

switch-to-telnet-console replace the console with a telnet console

Command Descriptions

<std-text-console>.info

Synopsis

<std-text-console>.info

Description

Print detailed information about the configuration of the device.

<std-text-console>.status

Synopsis

`<std-text-console>.status`

Description

Print detailed information about the current status of the device.

<std-text-console>.switch-to-host-serial-console**Synopsis**

`<std-text-console>.switch-to-host-serial-console ["port"]`

Description

Replace the std-text-console component with a std-host-serial-console component. The argument *port* is an existing COM port on Windows and a TTY on Linux. If no argument is given, a new pseudo port is opened. The name of the host device is returned.

See Also

[new-std-host-serial-console](#)

<std-text-console>.switch-to-telnet-console**Synopsis**

`<std-text-console>.switch-to-telnet-console [port]`

Description

Replace the %s component with a std-text-console component. The argument *port* is the port number that a telnet client can connect to. If no argument is given, a free port on the host is used. This port number is returned by the command.

See Also

[new-std-telnet-console](#)

std-text-graphics-console

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “std-text-graphics-console” component represents a text console for use with VGA instead of a graphics console.

Connectors

Name	Type	Direction
device	graphics-console	up
keyboard	keyboard	up

Attributes

bg_color

Optional attribute; **read/write** access; type: `string`. The background color.

fg_color

Optional attribute; **read/write** access; type: `string`. The foreground color.

instantiated

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: `string` or `nil`. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

title

Optional attribute; **read/write** access; type: `string`. The Window title.

top_component

Optional attribute; **read/write** access; type: `object` or `nil`. The top level component. Attribute is not valid until the component has been instantiated.

win32_font

Optional attribute; **read/write** access; type: `string`. Font to use in the console on Windows host.

window

Optional attribute; **read/write** access; type: `boolean`. Try to open window if TRUE (default). FALSE disables the window.

x11_font

Optional attribute; **read/write** access; type: `string`. Font to use in the console when using X11 (Linux/Solaris host).

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

<code>info</code>	print information about the device
<code>status</code>	print status of the device
<code>switch-to-graphics-console</code>	replace the console with a graphics console

Command Descriptions

`<std-text-graphics-console>.info`

Synopsis

`<std-text-graphics-console>.info`

Description

Print detailed information about the configuration of the device.

`<std-text-graphics-console>.status`

Synopsis

`<std-text-graphics-console>.status`

Description

Print detailed information about the current status of the device.

`<std-text-graphics-console>.switch-to-graphics-console`

Synopsis

<std-text-graphics-console>.switch-to-graphics-console

Description

Replace the std-text-graphics-console component with a std-graphics-console component.

See Also

[new-std-graphics-graphics-console](#)

std_mmc_card

Provided By
[std-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The std_mmc_card component represents an MMC/SD/SDHC/SDIO card.

Connectors

Name	Type	Direction
mmc_controller	mmc	up

Attributes

size

Required attribute; **read/write** access; type: `integer`. Card size, in bytes

Command List

Commands

[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

`<std_mmc_card>.info`

Synopsis

`<std_mmc_card>.info`

Description

Print detailed information about the configuration of the device.

`<std_mmc_card>.status`

Synopsis

`<std_mmc_card>.status`

Description

Print detailed information about the current status of the device.

std_sata_cdrom

Provided By

[std-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “sata_cdrom” component represents an Serial ATA CD-ROM.

Connectors

Name	Type	Direction
sata_slot	sata-slot	up

Command List

Commands

[**info**](#) print information about the device

[**status**](#) print status of the device

Command Descriptions

<std_sata_cdrom>.info

Synopsis

[**<std_sata_cdrom>.info**](#)

Description

Print detailed information about the configuration of the device.

<std_sata_cdrom>.status

Synopsis

[**<std_sata_cdrom>.status**](#)

Description

Print detailed information about the current status of the device.

std_sata_disk

Provided By
[std-components](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “sata_disk” component represents a Serial ATA Disk.

Connectors

Name	Type	Direction
sata_slot	sata-slot	up

Attributes

file

Optional attribute; **read/write** access; type: `string` or `nil`. File with disk contents for the full disk. Either a raw file or a CRAFF file.

size

Optional attribute; **read/write** access; type: `integer`. The size of the SATA disk in bytes. If it is omitted or less than the size of the image file, the disk size will be set to the same size as the image file.

Command List

Commands

[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

`<std_sata_disk>.info`

Synopsis

`<std_sata_disk>.info`

Description

Print detailed information about the configuration of the device.

`<std_sata_disk>.status`

Synopsis

`<std_sata_disk>.status`

Description

Print detailed information about the current status of the device.

top-component

Provided By
[Simics Core](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

Base top-level component class, should not be instantiated.

Attributes

components

Optional attribute; **read/write** access; type: [o*]. List of components below the top-level component. This attribute is not valid until the object has been instantiated.

cpu_list

Pseudo attribute; **read/write** access; type: [o*]. List of all processors below the top-level component. This attribute is not valid until the object has been instantiated.

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

machine_icon

Optional attribute; **read/write** access; type: string or nil. An instance of a top-level component may override the default *system_icon* with its own icon. This attribute is the name of an 80x80 pixel large icon in PNG format that should reside in the [host]/lib/images directory of the Simics installation or the workspace.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

system_info

Optional attribute; **read/write** access; type: string or nil. A short single-line description of the current configuration of the system that the component is a top-level of. The line may include the Unix name of the simulated machine, the installed operating system, or similar information. For example “Tango - Fedora Core 5 Linux”.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: `string`. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

system_icon

Pseudo class attribute; **read-only** access; type: `string` or `nil`. Name of an 80x80 pixels large icon in PNG format used to graphically represent the system that the component is a top-level of.

top_level

Pseudo class attribute; **read-only** access; type: `boolean` or `nil`. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

`<top-component>.info`

Synopsis

`<top-component>.info`

Description

Print detailed information about the configuration of the device.

`<top-component>.status`

Synopsis

`<top-component>.status`

Description

Print detailed information about the current status of the device.

usb-disk

Provided By
[usb-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “usb-disk” component represents an USB SCSI disk.

Connectors

Name	Type	Direction
usb-host	usb-port	up

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<usb-disk>.info

Synopsis

<usb-disk>.info

Description

Print detailed information about the configuration of the device.

<usb-disk>.status

Synopsis

<usb-disk>.status

Description

Print detailed information about the current status of the device.

usb-santa

Provided By
[usb-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “usb-santa” component represents an USB santa.

Connectors

Name	Type	Direction
usb-host	usb-port	up

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<usb-santa>.info

Synopsis

<usb-santa>.info

Description

Print detailed information about the configuration of the device.

<usb-santa>.status

Synopsis

<usb-santa>.status

Description

Print detailed information about the current status of the device.

usb-tablet-comp

Provided By
[usb-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “usb-tablet-comp” component represents a USB tablet device.

Connectors

Name	Type	Direction
usb-host	usb-port	up
abs-mouse	abs-mouse	down

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<usb-tablet-comp>.info

Synopsis

<usb-tablet-comp>.info

Description

Print detailed information about the configuration of the device.

<usb-tablet-comp>.status

Synopsis

<usb-tablet-comp>.status

Description

Print detailed information about the current status of the device.

usb-wacom-tablet-comp

Provided By
[usb-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The “usb-wacom-tablet-comp” component represents a USB Wacom tablet device.

Connectors

Name	Type	Direction
usb-host	usb-port	up
abs-mouse	abs-mouse	down

Attributes

instantiated

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the component has been instantiated.

object_prefix

Optional attribute; **read/write** access; type: string or nil. Object prefix string used by the component. The prefix is typically set by the **set-component-prefix** command before the component is created.

top_component

Optional attribute; **read/write** access; type: object or nil. The top level component. Attribute is not valid until the component has been instantiated.

Class Attributes

basename

Pseudo class attribute; **read-only** access; type: string. The basename of the component.

component_icon

Pseudo class attribute; **read-only** access; type: string or nil. Name of a 24x24 pixels large icon in PNG format used to graphically represent the component in a configuration viewer.

top_level

Pseudo class attribute; **read-only** access; type: boolean or nil. Set to TRUE for top-level components, i.e. the root of a hierarchy.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<usb-wacom-tablet-comp>.info

Synopsis

<usb-wacom-tablet-comp>.info

Description

Print detailed information about the configuration of the device.

<usb-wacom-tablet-comp>.status

Synopsis

<usb-wacom-tablet-comp>.status

Description

Print detailed information about the current status of the device.

usb_hs_keyboard_comp

Provided By
[usb-hid-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description
The High Speed USB Keyboard component class.

Connectors

Name	Type	Direction
connector_usb_host	usb-port	up
connector_keyboard	keyboard	down

Command List

Commands
[info](#) print information about the device
[status](#) print status of the device

Command Descriptions

<usb_hs_keyboard_comp>.info

Synopsis
`<usb_hs_keyboard_comp>.info`

Description
Print detailed information about the configuration of the device.

<usb_hs_keyboard_comp>.status

Synopsis
`<usb_hs_keyboard_comp>.status`

Description
Print detailed information about the current status of the device.

usb_keyboard_comp

Provided By
[usb-hid-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description
The USB Keyboard component class.

Connectors

Name	Type	Direction
connector_usb_host	usb-port	up
connector_keyboard	keyboard	down

Command List

Commands
[info](#) print information about the device
[status](#) print status of the device

Command Descriptions

<usb_keyboard_comp>.info

Synopsis
`<usb_keyboard_comp>.info`

Description
Print detailed information about the configuration of the device.

<usb_keyboard_comp>.status

Synopsis
`<usb_keyboard_comp>.status`

Description
Print detailed information about the current status of the device.

usb_mouse_comp

Provided By
[usb-hid-components](#)

Interfaces Implemented
[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description
The USB Mouse component class.

Connectors

Name	Type	Direction
connector_usb_host	usb-port	up
connector_abs_mouse	abs-mouse	down

Command List

Commands
[info](#) print information about the device
[status](#) print status of the device

Command Descriptions

<usb_mouse_comp>.info

Synopsis
`<usb_mouse_comp>.info`

Description
Print detailed information about the configuration of the device.

<usb_mouse_comp>.status

Synopsis
`<usb_mouse_comp>.status`

Description
Print detailed information about the current status of the device.

usb_xmas_tree

Provided By

[usb-comp](#)

Interfaces Implemented

[component](#), [component_connector](#), [conf_object](#), [log_object](#)

Description

The Xmas tree USB component.

Connectors

Name	Type	Direction
usb	usb-port	up

Command List

Commands

[**info**](#) print information about the device

[**status**](#) print status of the device

Command Descriptions

<usb_xmas_tree>.info

Synopsis

<usb_xmas_tree>.info

Description

Print detailed information about the configuration of the device.

<usb_xmas_tree>.status

Synopsis

<usb_xmas_tree>.status

Description

Print detailed information about the current status of the device.

Chapter 5

Classes

accel-vga

Provided By
[accel-vga](#)

Interfaces Implemented

`conf_object`, `image_snoop`, [io_memory](#), [log_object](#)

Description

Accelerated Super VGA device implementing the Bochs VBE protocol.

Attributes

console

Required attribute; **read/write** access; type: `object` or `nil`. Console object that must implement either `gfx_console` or both the `serial_device` and `extended_serial` interfaces.

direct_map_ram

Required attribute; **read/write** access; type: `object`. LFB memory object.

image

Required attribute; **read/write** access; type: `object`. Image object containing the VRAM.

memory_space

Required attribute; **read/write** access; type: `object`. Memory space to which the device is mapped. Needed for dynamic remapping.

Command List

Commands

info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
redraw	Redraw display
refresh-rate	Set rate at which accel-vga updated display
status	print status of the device
text-dump	Print text contents of display
wait-for-string	Wait for substring in text mode

Command Descriptions

`<accel-vga>.info`

Synopsis

`<accel-vga>.info`

Description

Print detailed information about the configuration of the device.

```
<accel-vga>.pci-header — deprecated
```

Synopsis

```
<accel-vga>.pci-header [-v]
```

Description

This command is deprecated; use [`<accel-vga>.print-pci-config-reg`](#)s instead.

```
<accel-vga>.print-pci-config-reg
```

Synopsis

```
<accel-vga>.print-pci-config-reg [-v]
```

Description

Print the PCI configuration registers. The `-v` flag turns on verbose mode.

```
<accel-vga>.redraw
```

Synopsis

```
<accel-vga>.redraw
```

Description

This command sends the current frame buffer contents of the simulated video device to the graphics console. If a simulated cursor is active, it is updated as well.

```
<accel-vga>.refresh-rate
```

Synopsis

```
<accel-vga>.refresh-rate [rate]
```

Description

Set the rate at which the accel-vga device updates the display.

The default is 1000 Hz (simulated). NOTE: The rate is currently the same for all accel-vga devices.

```
<accel-vga>.status
```

Synopsis

```
<accel-vga>.status
```

Description

Print detailed information about the current status of the device.

<accel-vga>.text-dump

Synopsis

<accel-vga>.text-dump

Description

Print the contents of the display in text mode.

<accel-vga>.wait-for-string

Synopsis

<accel-vga>.wait-for-string [-always] “*substring*” [*sample_interval*]

Description

Samples text mode output every *sample_interval* virtual seconds and returns if *substring* is present in the output. The *-always* flag is internal and should not be used.

AM79C973

Provided By
[AM79C973](#)

Interfaces Implemented

`conf_object, ieee_802_3_mac, ieee_802_3_mac_v3, io_memory, log_object`

Ports

`bcr (int_register, io_memory), csr (int_register, io_memory), ioreg (int_register, io_memory), pci_config (int_register, io_memory)`

Description

AM79C973 Ethernet controller.

Attributes

config_registers

Pseudo attribute; **read-only** access; type: `[i*]`. The PCI configuration registers, each 32 bits in size.

curr_rxd

Optional attribute; **read/write** access; type: `integer`. Index of the current receive descriptor

curr_txd

Optional attribute; **read/write** access; type: `integer`. Index of the current transmit descriptor

expansion_rom_size

Optional attribute; **read/write** access; type: `integer`. The size of the expansion ROM mapping.

irq_raised

Optional attribute; **read/write** access; type: `boolean`. Interrupt is currently raised by device

logical_address_filter

Optional attribute; **read/write** access; type: `integer`. The logical address filter

mac_address

Optional attribute; **read/write** access; type: `integer`. The MAC address

pci_bus

Required attribute; **read/write** access; type: `[os] or object`. The PCI bus this device is connected to, implementing the `pci-bus` interface.

pci_config_command

Optional attribute; **read/write** access; type: `integer`. The PCI command register.

pci_config_device_id

Optional attribute; **read/write** access; type: **integer**. The Device ID of the PCI device

pci_config_vendor_id

Optional attribute; **read/write** access; type: **integer**. The Vendor ID of the PCI device

phy

Optional attribute; **read/write** access; type: **[os], object, or nil**. The PHY the device is connected to.

poll_interval

Optional attribute; **read/write** access; type: **float**. Interval between transmit descriptor polls

recv_descr_tbl_addr

Optional attribute; **read/write** access; type: **integer**. The base address of the receive descriptor ring

recv_descr_tbl_length

Optional attribute; **read/write** access; type: **integer**. The length of the receive descriptor ring

xmt_descr_tbl_addr

Optional attribute; **read/write** access; type: **integer**. The base address of the transmit descriptor ring

xmt_descr_tbl_length

Optional attribute; **read/write** access; type: **integer**. The length of the transmit descriptor ring

Command List

Commands

info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
status	print status of the device

Command Descriptions

<AM79C973>.info

Synopsis

<AM79C973>.info

Description

Print detailed information about the configuration of the device.

<AM79C973>.pci-header — deprecated

Synopsis

<AM79C973>.pci-header [-v]

Description

This command is deprecated; use [**<AM79C973>.print-pci-config-reg**](#)s instead.

<AM79C973>.print-pci-config-reg

Synopsis

<AM79C973>.print-pci-config-reg [-v]

Description

Print the PCI configuration registers. The *-v* flag turns on verbose mode.

<AM79C973>.status

Synopsis

<AM79C973>.status

Description

Print detailed information about the current status of the device.

AT24Cxx

Provided By
[AT24Cxx](#)

Aliases
AT24C04

Interfaces Implemented
conf_object, [i2c_device](#), [i2c_slave_v2](#), [log_object](#)

Description
The AT24Cxx class implements Atmel's AT24C01A/02/04/08/16/32/64 serial EEPROM.

Attributes

current_address

Optional attribute; **read/write** access; type: integer. The current data address.

current_state

Optional attribute; **read/write** access; type: integer. The current state of the device.

i2c_address

Required attribute; **read/write** access; type: integer. I2C address of the device.

i2c_bus

Optional attribute; **read/write** access; type: [os], object, or nil. The I2C bus to which the device is connected.

memory

Required attribute; **read/write** access; type: data; **persistent** attribute. The on-chip memory bank.

page_size

Optional attribute; **read/write** access; type: integer. The size of the page.

Command List

Commands

[info](#) print information about the device
[status](#) print status of the device

Command Descriptions

<AT24Cxx>.info

Synopsis

<AT24Cxx>.info

Description

Print detailed information about the configuration of the device.

<AT24Cxx>.status

Synopsis

<AT24Cxx>.status

Description

Print detailed information about the current status of the device.

attr-meter

Provided By

[attr-meter](#)

Interfaces Implemented

[conf_object](#), [log_object](#)

Ports

derivative (scalar_time), value (scalar_time)

Description

Attribute meter

Attributes

realtime_period

Required attribute; **read/write** access; type: float or nil. Sample period (real time), in seconds, or nil to disable sampling

simtime_period

Required attribute; **read/write** access; type: float or nil. Sample period (simulated time), in seconds, or nil to disable sampling

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<attr-meter>.info

Synopsis

<attr-meter>.info

Description

Print detailed information about the configuration of the device.

<attr-meter>.status

Synopsis

<attr-meter>.status

Description

Print detailed information about the current status of the device.

base-trace-mem-hier

Provided By

[trace](#)

Interfaces Implemented

[conf_object](#), [log_object](#)

Description

The base class for the trace mode. This module provides an easy way of generating traces from Simics. Actions traced are executed instructions, memory accesses and, occurred exceptions. Traces will by default be printed as text to the terminal but can also be directed to a file in which case a binary format is available as well. It is also possible to control what will be traced by setting a few of the provided attributes.

Command List

Commands

- [**start**](#) control default tracer
- [**stop**](#) stop default tracer

Command Descriptions

`<base-trace-mem-hier>.start`

Alias

`<base-trace-mem-hier>.trace-start`

Synopsis

`<base-trace-mem-hier>.start [file] [-raw] [-same]`
`<base-trace-mem-hier>.stop`

Description

An installed tracer is required for this commands to work. Use the command [new-tracer](#) to get a default tracer.

The **start** command turns on the tracer. The tracer logs all executed instructions, memory accesses and exceptions. The output is written to *file* (appended) if given, otherwise to standard output. If the **-raw** flag is used, each trace entry is written in binary form; i.e., a `struct trace_entry`. This structure can be found in `trace.h`. Raw format can only be written to file.

If the **-same** flag is used, you will get the same output file and mode as the last time.

stop switches off the tracer and closes the file. Until you have given the **stop** command, you can not be sure that the entire trace has been written to the file.

See Also

[new-tracer](#)

<base-trace-mem-hier>.stop

Alias

<base-trace-mem-hier>.trace-stop

Synopsis

```
<base-trace-mem-hier>.stop  
<base-trace-mem-hier>.start [file] [-raw] [-same]
```

Description

An installed tracer is required for this commands to work. Use the command **new-tracer** to get a default tracer.

The **start** command turns on the tracer. The tracer logs all executed instructions, memory accesses and exceptions. The output is written to *file* (appended) if given, otherwise to standard output. If the **-raw** flag is used, each trace entry is written in binary form; i.e., a `struct trace_entry`. This structure can be found in `trace.h`. Raw format can only be written to file.

If the **-same** flag is used, you will get the same output file and mode as the last time.

stop switches off the tracer and closes the file. Until you have given the **stop** command, you can not be sure that the entire trace has been written to the file.

See Also

[new-tracer](#)

BCM5703C

Provided By
[BCM5703C](#)

Interfaces Implemented

`conf_object`, [ethernet_common](#), [ethernet_device](#), [io_memory](#), [log_object](#)

Description

The **BCM5703C** class models the Broadcom BCM5703C triple-speed 10/100/1000Base-T Ethernet LAN controller.

Attributes

add_crc_on_inject

Optional attribute; **read/write** access; type: `integer`. Frames injected using the ‘inject_packet’ will get a correctly calculated CRC added at the end when this attribute is set to 1 (default). When set to 0, the user has to supply a CRC field with the injected frame. Note that you must always provide room for the CRC field, even when this attribute is set to 1.

inject_packet

Pseudo attribute; **write-only** access; type: `data`. Attribute used to send a packet to the network device. Writing this attribute at any time injects a new packet into the device (without involving the network simulation). Injecting a packet copies the packet data, allowing the caller to reuse or dispose of the buffer used for creating the packet, after the attribute is written.

last_frame

Pseudo attribute; **read/write** access; type: `data` or `nil`. The frame that is currently about to be sent or received. The format is a raw Ethernet frame. This attribute is only valid in `Ethernet_Transmit` and `Ethernet_Receive` hap callbacks. It is possible to override the packet by assigning this attribute. The device will drop the current packet if the attribute is set to `Nil`, or with data of zero length.

link

Optional attribute; **read/write** access; type: `object` or `nil`. The Ethernet link that the network device is connected to.

mac_address

Optional attribute; **read/write** access; type: `[i{6}]`, `string`, or `nil`. Ethernet (MAC) address of the network interface.

model_crc

Optional attribute; **read/write** access; type: `integer`. If set to 1, the device will calculate and include an Ethernet CRC on all transmitted frames, and check the CRC on received frames. This attribute is **ignored** when new-style links are used.

tx_bandwidth

Optional attribute; **read/write** access; type: **integer**. The transmit bandwidth of the network interface in bits per second. The network interface will limit the rate at which it sends packets to remain below this bandwidth. Set to 0 for unlimited bandwidth.

Command List

Commands

connect	<i>deprecated</i> — connect to a simulated Ethernet link
disconnect	<i>deprecated</i> — disconnect from simulated link
info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
status	print status of the device

Command Descriptions

`<BCM5703C>.connect — deprecated`

Synopsis

`<BCM5703C>.connect [-auto] link`

Description

This command is deprecated; use [connect](#) instead.

Connect the device to a simulated Ethernet link. The flag '-auto' is deprecated and shouldn't be used.

See Also

[<BCM5703C>.disconnect](#)

`<BCM5703C>.disconnect — deprecated`

Synopsis

`<BCM5703C>.disconnect`

Description

This command is deprecated; use [disconnect](#) instead.

Disconnect the device from a simulated Ethernet link.

See Also

[<BCM5703C>.connect](#)

`<BCM5703C>.info`

Synopsis

`<BCM5703C>.info`

Description

Print detailed information about the configuration of the device.

<BCM5703C>.pci-header — deprecated

Synopsis

<BCM5703C>.pci-header [-v]

Description

This command is deprecated; use [**<BCM5703C>.print-pci-config-reg**](#)s instead.

<BCM5703C>.print-pci-config-reg

Synopsis

<BCM5703C>.print-pci-config-reg [-v]

Description

Print the PCI configuration registers. The *-v* flag turns on verbose mode.

<BCM5703C>.status

Synopsis

<BCM5703C>.status

Description

Print detailed information about the current status of the device.

BCM5704C

Provided By
[BCM5704C](#)

Interfaces Implemented

`conf_object`, [ethernet_common](#), [ethernet_device](#), [io_memory](#), [log_object](#)

Description

The **BCM5704C** class models the Broadcom BCM5704C triple-speed 10/100/1000Base-T Ethernet LAN controller.

Attributes

add_crc_on_inject

Optional attribute; **read/write** access; type: `integer`. Frames injected using the ‘inject_packet’ will get a correctly calculated CRC added at the end when this attribute is set to 1 (default). When set to 0, the user has to supply a CRC field with the injected frame. Note that you must always provide room for the CRC field, even when this attribute is set to 1.

inject_packet

Pseudo attribute; **write-only** access; type: `data`. Attribute used to send a packet to the network device. Writing this attribute at any time injects a new packet into the device (without involving the network simulation). Injecting a packet copies the packet data, allowing the caller to reuse or dispose of the buffer used for creating the packet, after the attribute is written.

is_mac1

Required attribute; **read/write** access; type: `integer`. Set to 1 if the device is the MAC1 in a dual MAC device

last_frame

Pseudo attribute; **read/write** access; type: `data` or `nil`. The frame that is currently about to be sent or received. The format is a raw Ethernet frame. This attribute is only valid in `Ethernet_Transmit` and `Ethernet_Receive` hap callbacks. It is possible to override the packet by assigning this attribute. The device will drop the current packet if the attribute is set to `Nil`, or with data of zero length.

link

Optional attribute; **read/write** access; type: `object` or `nil`. The Ethernet link that the network device is connected to.

mac_address

Optional attribute; **read/write** access; type: `[i{6}]`, `string`, or `nil`. Ethernet (MAC) address of the network interface.

model_crc

Optional attribute; **read/write** access; type: `integer`. If set to 1, the device will calculate and include an Ethernet CRC on all transmitted frames, and check the CRC on received frames. This attribute is **ignored** when new-style links are used.

other_bcm

Required attribute; **read/write** access; type: `object`. The other object forming a dual BCM device.

tx_bandwidth

Optional attribute; **read/write** access; type: `integer`. The transmit bandwidth of the network interface in bits per second. The network interface will limit the rate at which it sends packets to remain below this bandwidth. Set to 0 for unlimited bandwidth.

Command List

Commands

connect	<i>deprecated</i> — connect to a simulated Ethernet link
disconnect	<i>deprecated</i> — disconnect from simulated link
info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
status	print status of the device

Command Descriptions

`<BCM5704C>.connect — deprecated`

Synopsis

`<BCM5704C>.connect [-auto] link`

Description

This command is deprecated; use [connect](#) instead.

Connect the device to a simulated Ethernet link. The flag '-auto' is deprecated and shouldn't be used.

See Also

[<BCM5704C>.disconnect](#)

`<BCM5704C>.disconnect — deprecated`

Synopsis

`<BCM5704C>.disconnect`

Description

This command is deprecated; use [disconnect](#) instead.

Disconnect the device from a simulated Ethernet link.

See Also

[`<BCM5704C>.connect`](#)

<BCM5704C>.info**Synopsis**

`<BCM5704C>.info`

Description

Print detailed information about the configuration of the device.

<BCM5704C>.pci-header — deprecated**Synopsis**

`<BCM5704C>.pci-header [-v]`

Description

This command is deprecated; use [`<BCM5704C>.print-pci-config-reg`](#)s instead.

<BCM5704C>.print-pci-config-reg**Synopsis**

`<BCM5704C>.print-pci-config-reg [-v]`

Description

Print the PCI configuration registers. The `-v` flag turns on verbose mode.

<BCM5704C>.status**Synopsis**

`<BCM5704C>.status`

Description

Print detailed information about the current status of the device.

bitmask-translator

Provided By

[bitmask-translator](#)

Interfaces Implemented

`conf_object`, [log_object](#), `translate`

Description

This translator object will apply a bitmask (specified by the user) to all accesses. The target memory-space is responsible for actually having something mapped at the resulting address.

Attributes

address_mask

Required attribute; **read/write** access; type: `integer`. This mask will be applied to all accesses

Command List

Commands

[info](#) print information about the device

[status](#) print status of the device

Command Descriptions

<bitmask-translator>.info

Synopsis

`<bitmask-translator>.info`

Description

Print detailed information about the configuration of the device.

<bitmask-translator>.status

Synopsis

`<bitmask-translator>.status`

Description

Print detailed information about the current status of the device.

branch_recorder

Provided By

[Simics Core](#)

Interfaces Implemented

[address_profiler](#), [branch_arc](#), [conf_object](#), [log_object](#)

Description

A branch recorder records the branches taken by the zero or more processors that it is attached to. For each pair of addresses, the branch recorder keeps track of how many times the processors have branched from the first to the second address. This information can be used directly, or be processed to yield e.g. execution count for each address.

Attributes

address_type

Required attribute; **read/write** access; type: `integer`. Record physical or virtual addresses?

Command List

Commands defined by interface [address_profiler](#)

[address-profile-data](#), [address-profile-info](#), [address-profile-toplist](#)

breakpoints

Provided By
[Simics Core](#)

Interfaces Implemented
conf_object, [log_object](#)

Description

The set of breakpoints in Simics

cell

Provided By

[Simics Core](#)

Interfaces Implemented

[conf_object](#), [log_object](#)

Ports

`breakpoint_change` ([simple_dispatcher](#))

Description

A simulation cell, representing an autonomous partition of the configuration and able to be simulated in parallel with other cells. See the Multithreading chapter in the *Simics Model Builder User's Guide*.

Attributes

clocks

Pseudo attribute; **read-only** access; type: `[o*]`. Clock objects that belong to the cell.

Command List

Commands

[**cpu-switch-time**](#) get/set the time quantum for a given cell
[**info**](#) print information about the device

Command Descriptions

`<cell>.cpu-switch-time`

Synopsis

`<cell>.cpu-switch-time [cycles|seconds]`

Description

Change the time, in cycles or seconds, between CPU switches in the cell. Simics will simulate each processor for a specified number of cycles before switching to the next one. Specifying cycles (which is default) refers to the number of cycles on the first CPU in the cell. The following CPUs cycle switch times are calculated from their CPU frequencies. When issued with no argument, the current time quantum is reported.

See Also

[cpu-switch-time](#)

`<cell>.info`

Synopsis

`<cell>.info`

Description

Print detailed information about the configuration of the device.

central-client

Provided By
[central](#)

Interfaces Implemented
[conf_object](#), [log_object](#)

Description

This class handles communication and synchronization for distributed simulation.

Command List

Commands

connect	connect to Simics Central
disconnect	disconnect from Simics Central
info	print information about the device
links	list connected links

Command Descriptions

<central-client>.connect

Synopsis

<central-client>.connect ("server"|"obj")

Description

Connect Simics to a Simics Central server.

The *server* argument specifies the server to connect to. It is either of the form <addr> [:<port>] if a TCP connection should be used, or a file name if a file socket should be used.

To connect to a server object in the same Simics, use the *obj* argument instead.

See Also

[connect-central](#), [new-central-server](#), <central-client>.disconnect

<central-client>.disconnect

Synopsis

<central-client>.disconnect

Description

Disconnect from the Simics Central server.

This will make this Simics process standalone, and no longer part of a distributed simulation session.

See Also

[connect-central](#), [new-central-server](#), [<central-client>.connect](#)

<central-client>.info**Synopsis**

`<central-client>.info`

Description

Print detailed information about the configuration of the device.

<central-client>.links**Synopsis**

`<central-client>.links`

Description

List the link objects in this process that are shared in a distributed session.

When connected to a Simics Central, the link object in this list will be distributed.

central-server

Provided By
[central](#)

Interfaces Implemented
conf_object, [log_object](#)

Description

This class handles communication and synchronization for distributed simulation.

Command List

Commands

connections	list current connections
info	print information about the device
wait-for-clients	list current connections

Command Descriptions

<central-server>.connections

Synopsis

<central-server>.connections

Description

List all available connections, including the target, the type, and the current status of the connection.

See Also

[new-central-server](#)

<central-server>.info

Synopsis

<central-server>.info

Description

Print detailed information about the configuration of the device.

<central-server>.wait-for-clients

Synopsis

<central-server>.wait-for-clients [*clients*]

Description

Set the number of clients to wait for before letting the simulation start. Central does not care about already connected clients, it will wait until the correct number of new clients have been connected.

CL-PD6729

Provided By
[CL-PD6729](#)

Interfaces Implemented

`conf_object`, `io_memory`, `log_object`, `simple_interrupt`

Description

The CL-PD6729 class models a Cirrus Logic PD6729 PCI to PCMCIA bridge.

Command List

Commands

info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
status	print status of the device

Command Descriptions

`<CL-PD6729>.info`

Synopsis

`<CL-PD6729>.info`

Description

Print detailed information about the configuration of the device.

`<CL-PD6729>.pci-header — deprecated`

Synopsis

`<CL-PD6729>.pci-header [-v]`

Description

This command is deprecated; use [`<CL-PD6729>.print-pci-config-reg`](#) instead.

`<CL-PD6729>.print-pci-config-reg`

Synopsis

`<CL-PD6729>.print-pci-config-reg [-v]`

Description

Print the PCI configuration registers. The `-v` flag turns on verbose mode.

`<CL-PD6729>.status`

Synopsis

<CL-PD6729>.status

Description

Print detailed information about the current status of the device.

cli

Provided By

[Simics Core](#)

Interfaces Implemented

[conf_object](#), [log_object](#)

Description

Internal object for the command line

clipboard-gateway

Provided By
[clipboard-gateway](#)

Interfaces Implemented
conf_object, [log_object](#), [x86_cpuid](#)

Description
Clipboard gateway between host and target.

Attributes

clipboard_peer
Required attribute; **read/write** access; type: object. Clipboard peer

recorder
Required attribute; **read/write** access; type: object. History recorder

Command List

Commands
info print information about the device
status print status of the device

Command Descriptions

<clipboard-gateway>.info

Synopsis
<clipboard-gateway>.info

Description
Print detailed information about the configuration of the device.

<clipboard-gateway>.status

Synopsis
<clipboard-gateway>.status

Description
Print detailed information about the current status of the device.

clock

Provided By

[clock](#)

Interfaces Implemented

[conf_object](#), [cycle](#), [frequency_listener](#), [log_object](#)

Description

The **clock** class can be used to drive the simulation when there is no processor in the system. It implements the `cycle` interface, which means that devices and other objects can post events on it to be executed at a later time. The clock frequency determines the length of a clock cycle, which is used as the granularity for executing events. It is not possible to post step events to the clock.

Attributes

freq_mhz

Required attribute; **read/write** access; type: `float` or `integer`. Clock frequency in MHz.

Command List

Commands defined by interface [cycle](#)

[cycle-break](#), [cycle-break-absolute](#), [print-time](#), [wait-for-cycle](#), [wait-for-time](#)

Commands

[**info**](#) print information about the device

[**status**](#) print status of the device

Command Descriptions

<clock>.info

Synopsis

<clock>.info

Description

Print detailed information about the configuration of the device.

<clock>.status

Synopsis

<clock>.status

Description

Print detailed information about the current status of the device.

connector

Provided By
[Simics Core](#)

Interfaces Implemented
`conf_object`, [connector](#), [log_object](#)

Description

The connector object for connecting components.

Attributes

direction

Optional attribute; **read/write** access; type: `integer`. The direction of the connection. An *up* connector can only be connected to a *down* connector. An *any* connector can be connected to any other connector direction. Use the `Sim_Connector_Direction_*` constants when setting this attribute.

hotpluggable

Optional attribute; **read/write** access; type: `boolean`. The connector supports hot-plugging if `true`, not support hotplugging if `false`.

multi

Optional attribute; **read/write** access; type: `boolean`. The connector supports multiple connections if this attribute is `true`. Only one connection is supported if this attribute is `false`.

owner

Optional attribute; **read/write** access; type: `object`. The component object that is the owner of this connector.

required

Optional attribute; **read/write** access; type: `boolean`. The connector must be connected when instantiated if this attribute is `true`, otherwise `false`.

type

Optional attribute; **read/write** access; type: `string` or `nil`. The type of the connector as a string. Only two connectors of the same type can be connected.

Command List

Commands

[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

<connector>.info

Synopsis

<connector>.info

Description

Print detailed information about the configuration of the device.

<connector>.status

Synopsis

<connector>.status

Description

Print detailed information about the current status of the device.

context

Provided By

[Simics Core](#)

Interfaces Implemented

[breakpoint](#), [conf_object](#), [log_object](#)

Description

This is an abstraction of a virtual address space. It handles virtual-address breakpoints and may have symbol tables attached to it. Note that there is no automatic correspondence to any hardware or operating-system context, although such a mapping may be set up by the user.

Command List

Commands defined by interface [breakpoint](#)

[break](#), [tbreak](#)

Commands

finish-function	finish the current function
info	print information about the device
next-instruction	run to the next instruction, skipping subroutine calls
next-line	run to the next source line, skipping subroutine calls
off	switch off context object
on	switch on context object
reverse-next-instruction	reverse to the previous instruction, skipping subroutine calls
reverse-next-line	reverse to the previous source line, skipping subroutine calls
reverse-step-instruction	reverse to the previous instruction
reverse-step-line	reverse to the previous source line
reverse-until-activated	reverse until context becomes active
reverse-until-active	<i>deprecated</i> — reverse until context becomes active
reverse-until-deactivated	reverse until context becomes inactive
run-until-activated	run until context becomes active
run-until-active	<i>deprecated</i> — run until context becomes active
run-until-deactivated	run until context becomes inactive
status	print status of the device
step-instruction	run to the next instruction
step-line	run to the next source line
symtable	set the symbol table of a context
uncall-function	reverse to when the current function was called

Command Descriptions

`<context>.finish-function`

Alias

<context>.finish, <context>.fin

Synopsis

<context>.finish-function

Description

finish-function causes the simulation to run until the current function has returned.

See Also

[next-instruction](#), [next-line](#), [reverse-next-instruction](#), [reverse-next-line](#), [reverse-step-instruction](#), [reverse-step-line](#), [step-instruction](#), [step-line](#), [uncall-function](#)

<context>.info**Synopsis**

<context>.info

Description

Print detailed information about the configuration of the device.

<context>.next-instruction**Alias**

<context>.nexti, <context>.ni

Synopsis

<context>.next-instruction

Description

next-instruction causes the simulation to run until it reaches another instruction, but will not stop in subroutine calls.

See Also

[finish-function](#), [next-line](#), [reverse-next-instruction](#), [reverse-next-line](#), [reverse-step-instruction](#), [reverse-step-line](#), [step-instruction](#), [step-line](#), [uncall-function](#)

<context>.next-line**Alias**

<context>.next, <context>.n

Synopsis

<context>.next-line

Description

next-line causes the simulation to run until it reaches another source line, but will not stop in subroutine calls.

See Also

[finish-function](#), [next-instruction](#), [reverse-next-instruction](#), [reverse-next-line](#), [reverse-step-instruction](#), [reverse-step-line](#), [step-instruction](#), [step-line](#), [uncall-function](#)

<context>.off

Synopsis

```
<context>.off  
<context>.on
```

Description

`<context>.on` activates the effects of a context object, i.e., breakpoints on virtual addresses. `<context>.off` deactivates a context object.

<context>.on

Synopsis

```
<context>.on  
<context>.off
```

Description

`<context>.on` activates the effects of a context object, i.e., breakpoints on virtual addresses. `<context>.off` deactivates a context object.

<context>.reverse-next-instruction

Alias

`<context>.rnexti`, `<context>.rni`

Synopsis

```
<context>.reverse-next-instruction
```

Description

reverse-next-instruction causes the simulation to run backwards until it reaches another instruction, but will not stop in subroutine calls.

See Also

[finish-function](#), [next-instruction](#), [next-line](#), [reverse-next-line](#), [reverse-step-instruction](#), [reverse-step-line](#), [step-instruction](#), [step-line](#), [uncall-function](#)

<context>.reverse-next-line

Alias

`<context>.rnext`, `<context>.rn`

Synopsis

```
<context>.reverse-next-line
```

Description

reverse-next-line causes the simulation to run backwards until it reaches another source line, but will not stop in subroutine calls.

See Also

[finish-function](#), [next-instruction](#), [next-line](#), [reverse-next-instruction](#), [reverse-step-instruction](#), [reverse-step-line](#), [step-instruction](#), [step-line](#), [uncall-function](#)

<context>.reverse-step-instruction

Alias

<context>.rstepi, <context>.rsi, <context>.unstep-instruction, <context>.ui

Synopsis

<context>.reverse-step-instruction

Description

reverse-step-instruction causes the simulation to run one instruction in reverse.

See Also

[finish-function](#), [next-instruction](#), [next-line](#), [reverse-next-instruction](#), [reverse-next-line](#), [reverse-step-line](#), [step-instruction](#), [step-line](#), [uncall-function](#)

<context>.reverse-step-line

Alias

<context>.rstep, <context>.rs

Synopsis

<context>.reverse-step-line

Description

reverse-step-line causes the simulation to run in reverse until it reaches another source line.

See Also

[finish-function](#), [next-instruction](#), [next-line](#), [reverse-next-instruction](#), [reverse-next-line](#), [reverse-step-instruction](#), [step-instruction](#), [step-line](#), [uncall-function](#)

<context>.reverse-until-activated

Synopsis

<context>.reverse-until-activated

Description

Reverse until the context is activated on a processor. (For example: if the context was last activated at time 8, then deactivated at time 13, running this command at time 21 will reverse to time 8.)

See Also

[`<context>.run-until-activated`](#), [`<context>.run-until-deactivated`](#), [`<context>.reverse-until-deactivated`](#)

<context>.reverse-until-active — deprecated

Synopsis

`<context>.reverse-until-active`

Description

This command is deprecated; use [`reverse-until-activated`](#) instead.

This is a deprecated alias for [`<context>.reverse-until-activated`](#)

<context>.reverse-until-deactivated

Synopsis

`<context>.reverse-until-deactivated`

Description

Reverse until the context is deactivated on a processor. (For example: if the context was last activated at time 8, then deactivated at time 13, running this command at time 21 will reverse to time 13.)

See Also

[`<context>.run-until-activated`](#), [`<context>.reverse-until-activated`](#), [`<context>.run-until-deactivated`](#)

<context>.run-until-activated

Synopsis

`<context>.run-until-activated`

Description

Run until the context is activated on a processor.

See Also

[`<context>.reverse-until-activated`](#), [`<context>.run-until-deactivated`](#), [`<context>.reverse-until-deactivated`](#)

<context>.run-until-active — deprecated

Synopsis

`<context>.run-until-active`

Description

This command is deprecated; use [`run-until-activated`](#) instead.

This is a deprecated alias for [`<context>.run-until-activated`](#)

<context>.run-until-deactivated

Synopsis

<context>.run-until-deactivated

Description

Run until the context is deactivated on a processor.

See Also

[**<context>.run-until-activated**](#), [**<context>.reverse-until-activated**](#), [**<context>.reverse-until-deactivated**](#)

<context>.status

Synopsis

<context>.status

Description

Print detailed information about the current status of the device.

<context>.step-instruction

Alias

<context>.stepi, **<context>.si**

Synopsis

<context>.step-instruction

Description

step-instruction causes to simulation to run one instruction.

See Also

[**finish-function**](#), [**next-instruction**](#), [**next-line**](#), [**reverse-next-instruction**](#), [**reverse-next-line**](#), [**reverse-step-instruction**](#), [**reverse-step-line**](#), [**step-line**](#), [**uncall-function**](#)

<context>.step-line

Alias

<context>.step, **<context>.s**

Synopsis

<context>.step-line

Description

step-line causes the simulation to run until it reaches another source line.

See Also

[**finish-function**](#), [**next-instruction**](#), [**next-line**](#), [**reverse-next-instruction**](#), [**reverse-next-line**](#), [**reverse-step-instruction**](#), [**reverse-step-line**](#), [**step-instruction**](#), [**uncall-function**](#)

<context>.symtable

Synopsis

```
<context>.symtable ["symtable"]
```

Description

Sets the symbol table of the context to *symtable*, or displays the name of the symtable currently bound to the context. The symbol table is created if it does not already exist.

See Also

[set-context](#), [new-symtable](#)

<context>.uncall-function

Alias

```
<context>.uncall
```

Synopsis

```
<context>.uncall-function
```

Description

uncall-function causes the simulation to run backwards until just before the current function was called.

See Also

[finish-function](#), [next-instruction](#), [next-line](#), [reverse-next-instruction](#), [reverse-next-line](#), [reverse-step-instruction](#), [reverse-step-line](#), [step-instruction](#), [step-line](#)

coverage_profiler

Provided By
[os-awareness](#)

Interfaces Implemented
conf_object, [log_object](#)

Description

Objects of the **coverage_profiler** class are created by the `(os_awareness).code-coverage` command. They can be used to save the formatted coverage profile.

Attributes

process_name

Required attribute; **read/write** access; type: string. The process name to wait for.

syntable

Required attribute; **read/write** access; type: object. The syntable object.

tracker

Required attribute; **read/write** access; type: object. The software tracker used to track software.

Command List

Commands

info print information about the device

save save the coverage profile

status print status of the device

Command Descriptions

<coverage_profiler>.info

Synopsis

`<coverage_profiler>.info`

Description

Print detailed information about the configuration of the device.

<coverage_profiler>.save

Synopsis

`<coverage_profiler>.save [output] [format]`

Description

Format and save the collected coverage data.

The coverage profile will be written to *output*. The output format is selected with *format*. The default is `html`, which will create a directory of HTML files. The `raw` format is useful for postprocessing to create other formats by external means. The default output name is based on the process name.

```
<coverage_profiler>.status
```

Synopsis

```
<coverage_profiler>.status
```

Description

Print detailed information about the current status of the device.

cpu-group

Provided By

[cpu-group](#)

Interfaces Implemented

[conf_object](#), [log_object](#)

Description

The cpu-group class groups processors together. For example processors that share memory and/or can interrupt each other.

Attributes

cpu_list

Required attribute; **read/write** access; type: [o*]. List of all connected processors. This attribute is available in all classes implementing the “cpu_group” interface.

Command List

Commands

[info](#) print information about the device

Command Descriptions

<cpu-group>.info

Synopsis

<cpu-group>.info

Description

Print detailed information about the configuration of the device.

data-profiler

Provided By

[Simics Core](#)

Interfaces Implemented

[address_profiler](#), [conf_object](#), [log_object](#)

Description

A data profiler maintains one counter for every interval of addresses that differ only in the last granularity bits. This is used for various profiling tasks, for example counting the number of cache misses in each of the intervals.

Command List

Commands defined by interface [address_profiler](#)

[address-profile-data](#), [address-profile-info](#), [address-profile-toplist](#)

Commands

[clear](#) clear data profiler

Command Descriptions

<data-profiler>.clear

Synopsis

[<data-profiler>.clear](#)

Description

Reset all counters of the data profiler to zero.

datagram_link_endpoint

Provided By

[datagram-link](#)

Interfaces Implemented

`conf_object`, [datagram_link](#), [log_object](#)

Description

Endpoint for datagram_link objects.

Attributes

device

Required attribute; **read/write** access; type: `[os], object, or nil`. The device connected to this endpoint.

id

Required attribute; **read/write** access; type: `integer`. Endpoint ID. The ID of each endpoint must be unique among the link's endpoints, and it may not be 0 or `0xffffffffffff`

link

Required attribute; **read/write** access; type: `object`. The link object to which this endpoint belongs.

Command List

Commands

info print information about the device

status print status of the device

Command Descriptions

<datagram_link_endpoint>.info

Synopsis

`<datagram_link_endpoint>.info`

Description

Print detailed information about the configuration of the device.

<datagram_link_endpoint>.status

Synopsis

`<datagram_link_endpoint>.status`

Description

Print detailed information about the current status of the device.

datagram_link_impl

Provided By
[datagram-link](#)

Interfaces Implemented
conf_object, [log_object](#)

Description
A link that broadcasts byte strings.

Attributes

goal_latency
Required attribute; **read/write** access; type: **float**. The desired latency for this link.

Command List

Commands
info print information about the device
status print status of the device

Command Descriptions

<datagram_link_impl>.info

Synopsis
<datagram_link_impl>.info

Description
Print detailed information about the configuration of the device.

<datagram_link_impl>.status

Synopsis
<datagram_link_impl>.status

Description
Print detailed information about the current status of the device.

DEC21041

Provided By
[DEC21041](#)

Interfaces Implemented

`conf_object, ethernet_common, ethernet_device, io_memory, log_object`

Description

The DEC21041 is a fast Ethernet LAN controller providing a direct interface to the PCI bus. The DEC21041 interfaces to the host processor by using onchip control and status registers (CSRs) and a shared host memory area, set up mainly during initialization.

Current limitations:

- Inquiry accesses are not supported.
- Only aligned, 32-bits CSR accesses are supported.
- Large incoming frames will not be split over several buffers, but dropped.
- Mac address filtering is not supported.

Attributes

add_crc_on_inject

Optional attribute; **read/write** access; type: `integer`. Frames injected using the ‘inject_packet’ will get a correctly calculated CRC added at the end when this attribute is set to 1 (default). When set to 0, the user has to supply a CRC field with the injected frame. Note that you must always provide room for the CRC field, even when this attribute is set to 1.

inject_packet

Pseudo attribute; **write-only** access; type: `data`. Attribute used to send a packet to the network device. Writing this attribute at any time injects a new packet into the device (without involving the network simulation). Injecting a packet copies the packet data, allowing the caller to reuse or dispose of the buffer used for creating the packet, after the attribute is written.

last_frame

Pseudo attribute; **read/write** access; type: `data` or `nil`. The frame that is currently about to be sent or received. The format is a raw Ethernet frame. This attribute is only valid in `Ethernet_Transmit` and `Ethernet_Receive` hap callbacks. It is possible to override the packet by assigning this attribute. The device will drop the current packet if the attribute is set to `Nil`, or with data of zero length.

link

Optional attribute; **read/write** access; type: `object` or `nil`. The Ethernet link that the network device is connected to.

mac_address

Optional attribute; **read/write** access; type: [i{6}], string, or nil. Ethernet (MAC) address of the network interface.

model_crc

Optional attribute; **read/write** access; type: integer. If set to 1, the device will calculate and include an Ethernet CRC on all transmitted frames, and check the CRC on received frames. This attribute is **ignored** when new-style links are used.

Command List

Commands

connect	<i>deprecated</i> — connect to a simulated Ethernet link
disconnect	<i>deprecated</i> — disconnect from simulated link
info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
status	print status of the device

Command Descriptions

`<DEC21041>.connect — deprecated`

Synopsis

`<DEC21041>.connect [-auto] link`

Description

This command is deprecated; use [connect](#) instead.

Connect the device to a simulated Ethernet link. The flag '-auto' is deprecated and shouldn't be used.

See Also

[<DEC21041>.disconnect](#)

`<DEC21041>.disconnect — deprecated`

Synopsis

`<DEC21041>.disconnect`

Description

This command is deprecated; use [disconnect](#) instead.

Disconnect the device from a simulated Ethernet link.

See Also

[<DEC21041>.connect](#)

<DEC21041>.info

Synopsis

<DEC21041>.info

Description

Print detailed information about the configuration of the device.

<DEC21041>.pci-header — *deprecated*

Synopsis

<DEC21041>.pci-header [-v]

Description

This command is deprecated; use [**<DEC21041>.print-pci-config-reg**](#)s instead.

<DEC21041>.print-pci-config-reg

Synopsis

<DEC21041>.print-pci-config-reg [-v]

Description

Print the PCI configuration registers. The *-v* flag turns on verbose mode.

<DEC21041>.status

Synopsis

<DEC21041>.status

Description

Print detailed information about the current status of the device.

DEC21140A

Provided By
[DEC21140A](#)

Interfaces Implemented

`conf_object`, [ethernet_common](#), [ethernet_device](#), [io_memory](#), [log_object](#)

Description

The DEC21140A is a fast Ethernet LAN controller providing a direct interface to the PCI bus. The DEC21140A interfaces to the host processor by using onchip control and status registers (CSRs) and a shared host memory area, set up mainly during initialization.

Current limitations:

- Inquiry accesses are not supported.
- Only aligned, 32-bits CSR accesses are supported.
- Large incoming frames will not be split over several buffers, but dropped.
- Mac address filtering is not supported.

Attributes

add_crc_on_inject

Optional attribute; **read/write** access; type: `integer`. Frames injected using the ‘inject_packet’ will get a correctly calculated CRC added at the end when this attribute is set to 1 (default). When set to 0, the user has to supply a CRC field with the injected frame. Note that you must always provide room for the CRC field, even when this attribute is set to 1.

inject_packet

Pseudo attribute; **write-only** access; type: `data`. Attribute used to send a packet to the network device. Writing this attribute at any time injects a new packet into the device (without involving the network simulation). Injecting a packet copies the packet data, allowing the caller to reuse or dispose of the buffer used for creating the packet, after the attribute is written.

last_frame

Pseudo attribute; **read/write** access; type: `data` or `nil`. The frame that is currently about to be sent or received. The format is a raw Ethernet frame. This attribute is only valid in `Ethernet_Transmit` and `Ethernet_Receive` hap callbacks. It is possible to override the packet by assigning this attribute. The device will drop the current packet if the attribute is set to `Nil`, or with data of zero length.

link

Optional attribute; **read/write** access; type: `object` or `nil`. The Ethernet link that the network device is connected to.

mac_address

Optional attribute; **read/write** access; type: [i{6}], string, or nil. Ethernet (MAC) address of the network interface.

model_crc

Optional attribute; **read/write** access; type: integer. If set to 1, the device will calculate and include an Ethernet CRC on all transmitted frames, and check the CRC on received frames. This attribute is **ignored** when new-style links are used.

Command List

Commands

connect	<i>deprecated</i> — connect to a simulated Ethernet link
disconnect	<i>deprecated</i> — disconnect from simulated link
info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
status	print status of the device

Command Descriptions

`<DEC21140A>.connect — deprecated`

Synopsis

`<DEC21140A>.connect [-auto] link`

Description

This command is deprecated; use [connect](#) instead.

Connect the device to a simulated Ethernet link. The flag '-auto' is deprecated and shouldn't be used.

See Also

[<DEC21140A>.disconnect](#)

`<DEC21140A>.disconnect — deprecated`

Synopsis

`<DEC21140A>.disconnect`

Description

This command is deprecated; use [disconnect](#) instead.

Disconnect the device from a simulated Ethernet link.

See Also

[<DEC21140A>.connect](#)

<DEC21140A>.info

Synopsis

<DEC21140A>.info

Description

Print detailed information about the configuration of the device.

<DEC21140A>.pci-header — *deprecated*

Synopsis

<DEC21140A>.pci-header [-v]

Description

This command is deprecated; use [**<DEC21140A>.print-pci-config-reg**](#)s instead.

<DEC21140A>.print-pci-config-reg

Synopsis

<DEC21140A>.print-pci-config-reg [-v]

Description

Print the PCI configuration registers. The *-v* flag turns on verbose mode.

<DEC21140A>.status

Synopsis

<DEC21140A>.status

Description

Print detailed information about the current status of the device.

DEC21140A-dml

Provided By
[DEC21140A-dml](#)

Interfaces Implemented

`conf_object, ieee_802_3_mac, ieee_802_3_mac_v3, io_memory, log_object`

Ports

`csr (int_register, io_memory), pci_config (int_register, io_memory)`

Description

DEC21140A Ethernet controller.

Attributes

config_registers

Pseudo attribute; **read-only** access; type: `[i*]`. The PCI configuration registers, each 32 bits in size.

expansion_rom_size

Optional attribute; **read/write** access; type: `integer`. The size of the expansion ROM mapping.

mii_bus

Required attribute; **read/write** access; type: `[os] or object`. MII Management Bus

pci_bus

Required attribute; **read/write** access; type: `[os] or object`. The PCI bus this device is connected to, implementing the `pci-bus` interface.

pci_config_command

Optional attribute; **read/write** access; type: `integer`. The PCI command register.

pci_config_device_id

Optional attribute; **read/write** access; type: `integer`. The Device ID of the PCI device

pci_config_vendor_id

Optional attribute; **read/write** access; type: `integer`. The Vendor ID of the PCI device

phy

Required attribute; **read/write** access; type: `[os] or object`. Phy object

serial_eeprom

Required attribute; **read/write** access; type: `[os] or object`. Serial EEPROM

Command List

Commands

pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers

Command Descriptions

`<DEC21140A-dml>.pci-header — deprecated`

Synopsis

`<DEC21140A-dml>.pci-header [-v]`

Description

This command is deprecated; use [`<DEC21140A-dml>.print-pci-config-reg`](#) instead.

`<DEC21140A-dml>.print-pci-config-reg`

Synopsis

`<DEC21140A-dml>.print-pci-config-reg [-v]`

Description

Print the PCI configuration registers. The `-v` flag turns on verbose mode.

DEC21143

Provided By
[DEC21143](#)

Interfaces Implemented

`conf_object`, [ethernet_common](#), [ethernet_device](#), [io_memory](#), [log_object](#)

Description

The DEC21143 is a fast Ethernet LAN controller providing a direct interface to the PCI bus. The DEC21143 interfaces to the host processor by using onchip control and status registers (CSRs) and a shared host memory area, set up mainly during initialization.

Current limitations:

- Inquiry accesses are not supported.
- Only aligned, 32-bits CSR accesses are supported.
- Large incoming frames will not be split over several buffers, but dropped.
- Mac address filtering is not supported.

Attributes

add_crc_on_inject

Optional attribute; **read/write** access; type: `integer`. Frames injected using the ‘inject_packet’ will get a correctly calculated CRC added at the end when this attribute is set to 1 (default). When set to 0, the user has to supply a CRC field with the injected frame. Note that you must always provide room for the CRC field, even when this attribute is set to 1.

inject_packet

Pseudo attribute; **write-only** access; type: `data`. Attribute used to send a packet to the network device. Writing this attribute at any time injects a new packet into the device (without involving the network simulation). Injecting a packet copies the packet data, allowing the caller to reuse or dispose of the buffer used for creating the packet, after the attribute is written.

last_frame

Pseudo attribute; **read/write** access; type: `data` or `nil`. The frame that is currently about to be sent or received. The format is a raw Ethernet frame. This attribute is only valid in `Ethernet_Transmit` and `Ethernet_Receive` hap callbacks. It is possible to override the packet by assigning this attribute. The device will drop the current packet if the attribute is set to `Nil`, or with data of zero length.

link

Optional attribute; **read/write** access; type: `object` or `nil`. The Ethernet link that the network device is connected to.

mac_address

Optional attribute; **read/write** access; type: [i{6}], string, or nil. Ethernet (MAC) address of the network interface.

model_crc

Optional attribute; **read/write** access; type: integer. If set to 1, the device will calculate and include an Ethernet CRC on all transmitted frames, and check the CRC on received frames. This attribute is **ignored** when new-style links are used.

Command List

Commands

connect	<i>deprecated</i> — connect to a simulated Ethernet link
disconnect	<i>deprecated</i> — disconnect from simulated link
info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
status	print status of the device

Command Descriptions

`<DEC21143>.connect — deprecated`

Synopsis

`<DEC21143>.connect [-auto] link`

Description

This command is deprecated; use [connect](#) instead.

Connect the device to a simulated Ethernet link. The flag '-auto' is deprecated and shouldn't be used.

See Also

[<DEC21143>.disconnect](#)

`<DEC21143>.disconnect — deprecated`

Synopsis

`<DEC21143>.disconnect`

Description

This command is deprecated; use [disconnect](#) instead.

Disconnect the device from a simulated Ethernet link.

See Also

[<DEC21143>.connect](#)

<DEC21143>.info

Synopsis

<DEC21143>.info

Description

Print detailed information about the configuration of the device.

<DEC21143>.pci-header — *deprecated*

Synopsis

<DEC21143>.pci-header [-v]

Description

This command is deprecated; use [**<DEC21143>.print-pci-config-reg**](#)s instead.

<DEC21143>.print-pci-config-reg

Synopsis

<DEC21143>.print-pci-config-reg [-v]

Description

Print the PCI configuration registers. The *-v* flag turns on verbose mode.

<DEC21143>.status

Synopsis

<DEC21143>.status

Description

Print detailed information about the current status of the device.

disassemble_v9

Provided By

[disassemble_v9](#)

Interfaces Implemented

conf_object, disassemble, [log_object](#)

Description

This class implements the disassemble interface for the v9 architecture. Use the attributes to set up the operating mode prior to using the interface.

disassemble_x86

Provided By

[disassemble_x86](#)

Interfaces Implemented

`conf_object`, `disassemble`, [log_object](#)

Description

This class implements the disassemble interface for the x86 architecture. Use the attributes to set up the operating mode prior to using the interface.

dm9161

Provided By
[dm9161](#)

Interfaces Implemented

`conf_object`, `ethernet_common`, `ieee_802_3_phy`, `ieee_802_3_phy_v2`, `ieee_802_3_phy_v3`,
`log_object`, `mii`, `mii_management`

Ports

`mii_regs` ([int_register](#))

Description

DAVICOM DM9161 10/100M PHY Transceiver

Attributes

address

Optional attribute; **read/write** access; type: `integer`. PHY identifier sent in calls to `ieee_802_3_mac` interface methods.

link

Optional attribute; **read/write** access; type: `[os], object, or nil`. The Ethernet link that the PHY is connected to or Nil if the PHY is disconnected. Must implement the `ethernet_common` interface.

link_led

Optional attribute; **read/write** access; type: `[os], object, or nil`. An object implementing the signal interface. It will ge a high signal when the link status is up.

loopback_delay

Optional attribute; **read/write** access; type: `float`. The time in seconds for sending back a frame when in loopback mode

mac

Optional attribute; **read/write** access; type: `[os], object, or nil`. Media access controller (MAC) object, implemenitng the `ieee_802_3_mac_v3` interface.

mii_regs_ext_status

Optional attribute; **read/write** access; type: `integer`. Default value of MII Extended Status (MII register address 15)

mii_regs_status

Optional attribute; **read/write** access; type: `integer`. Default value of MII Status (MII register address 1)

phy_id

Pseudo attribute; **read/write** access; type: `integer`. 32-bit PHY identifier

register_defaults

Optional attribute; **read/write** access; type: [n|i{32}]. Reset values of the MII registers. A **nil** value denotes that the standard reset value for that register is used.

registers

Pseudo attribute; **read/write** access; type: [i{32}]. The 32 MII management registers

tx_bandwidth

Optional attribute; **read/write** access; type: **integer**. The transmit bandwidth of the network interface in bits per second. The network interface will limit the rate at which it sends packets to remain below this bandwidth. Set to 0 for unlimited bandwidth.

Command List

Commands

info print information about the device

status print status of the device

Command Descriptions

<dm9161>.info

Synopsis

<dm9161>.info

Description

Print detailed information about the configuration of the device.

<dm9161>.status

Synopsis

<dm9161>.status

Description

Print detailed information about the current status of the device.

dynamic_link_connector

Provided By

[Simics Core](#)

Interfaces Implemented

`conf_object, connector, log_object`

Description

Dynamic link connector representing one connection to the link. Everytime this connector is filled, it tries---if it is allowed---to create a new instance of itself so there is always one free connector available. It is useful for links that accept an undetermined number of connections (although the component itself may set limit to how this connector can grow).

Attributes

connector_name

Required attribute; **read/write** access; type: `string`. Name of the connector

connector_template

Required attribute; **read/write** access; type: `string`. Name of the link connector template used to create this connector

connector_type

Required attribute; **read/write** access; type: `string`. Type used to match which other connector objects this connector can connect to.

direction

Required attribute; **read/write** access; type: `integer`. Direction of the connector: up, down or any

hotpluggable

Required attribute; **read/write** access; type: `boolean`. If true, this connector can be connected or disconnected after instantiation. If false, the connection must be made before the component is instantiated.

multi

Required attribute; **read/write** access; type: `boolean`. If true, more than one connector object can be connected to this connector at the same time.

owner

Required attribute; **read/write** access; type: `object`. Component to which this connector applies

required

Required attribute; **read/write** access; type: `boolean`. If true, this connector should be connected before its component can be instantiated. If false, it can be left empty.

slot_template

Required attribute; **read/write** access; type: `string`. Name template to be used for newly created connectors. The name template is expected to contain at least one '%d' that will be used to give the connector a unique number.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

`<dynamic_link_connector>.info`

Synopsis

`<dynamic_link_connector>.info`

Description

Print detailed information about the configuration of the device.

`<dynamic_link_connector>.status`

Synopsis

`<dynamic_link_connector>.status`

Description

Print detailed information about the current status of the device.

empty-device-c

Provided By
[empty-device-c](#)

Interfaces Implemented
conf_object, [io_memory](#), [log_object](#)

Description
This is a long description of this class.

Command List

Commands
[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

<empty-device-c>.info

Synopsis
<empty-device-c>.info

Description
Print detailed information about the configuration of the device.

<empty-device-c>.status

Synopsis
<empty-device-c>.status

Description
Print detailed information about the current status of the device.

empty-device-cc

Provided By

[empty-device-cc](#)

Interfaces Implemented

`conf_object, io_memory, log_object`

Description

This is a documentation string describing the empty-device-cc class.

Command List

Commands

`info` print information about the device
`status` print status of the device

Command Descriptions

`<empty-device-cc>.info`

Synopsis

`<empty-device-cc>.info`

Description

Print detailed information about the configuration of the device.

`<empty-device-cc>.status`

Synopsis

`<empty-device-cc>.status`

Description

Print detailed information about the current status of the device.

etg

Provided By

[etg](#)

Interfaces Implemented

conf_object, [ethernet_cable](#), [ethernet_common](#), [ethernet_device](#), [log_object](#)

Ports

packet_size ([uint64_state](#)), pps ([uint64_state](#)), start ([signal](#))

Description

The ethernet-traffic-generator (etg) device emulates a network adapter and generates traffic.

Attributes

dst_ip

Required attribute; **read/write** access; type: `string`. IP address to send packets to.

ip

Required attribute; **read/write** access; type: `string`. IP address assigned to the device.

netmask

Required attribute; **read/write** access; type: `integer or string`. Netmask defining the subnet for the device.

Command List

Commands

info	print information about the device
packet-rate	Set or display the packets per second rate
packet-size	Set or display the packet size
start	Start generating traffic
status	print status of the device
stop	Stop generating traffic

Command Descriptions

`<etg>.info`

Synopsis

`<etg>.info`

Description

Print detailed information about the configuration of the device.

`<etg>.packet-rate`

Synopsis

`<etg>.packet-rate [rate]`

Description

Set the rate, in packets per second, at which packets are sent by the traffic generator.

If no rate is given, the current rate is displayed.

`<etg>.packet-size`

Synopsis

`<etg>.packet-size [size]`

Description

Set the size of the packets that are sent by the traffic generator.

If no size is given, the current size is displayed.

`<etg>.start`

Synopsis

`<etg>.start [count]`

Description

Start the traffic generator.

`<etg>.status`

Synopsis

`<etg>.status`

Description

Print detailed information about the current status of the device.

`<etg>.stop`

Synopsis

`<etg>.stop`

Description

Stop the traffic generator.

eth-cable-link

Provided By
[eth-links](#)

Interfaces Implemented
conf_object, [ethernet_snoop](#), [log_object](#)

Description
Ethernet cable link

Attributes

goal_latency
Required attribute; **read/write** access; type: float. The desired latency for this link.

Command List

Commands
[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

<eth-cable-link>.info

Synopsis
<eth-cable-link>.info

Description
Print detailed information about the configuration of the device.

<eth-cable-link>.status

Synopsis
<eth-cable-link>.status

Description
Print detailed information about the current status of the device.

eth-cable-link-endpoint

Provided By

[eth-links](#)

Interfaces Implemented

`conf_object`, [ethernet_cable](#), [ethernet_common](#), [log_object](#)

Description

Ethernet cable link endpoint

Attributes

device

Required attribute; **read/write** access; type: `[os], object, or nil`. The device connected to this endpoint.

id

Required attribute; **read/write** access; type: `integer`. Endpoint ID. The ID of each endpoint must be unique among the link's endpoints, and it may not be 0 or `0xffffffffffff`

link

Required attribute; **read/write** access; type: `object`. The link object to which this endpoint belongs.

Command List

Commands

[**info**](#) print information about the device

[**status**](#) print status of the device

Command Descriptions

`<eth-cable-link-endpoint>.info`

Synopsis

`<eth-cable-link-endpoint>.info`

Description

Print detailed information about the configuration of the device.

`<eth-cable-link-endpoint>.status`

Synopsis

`<eth-cable-link-endpoint>.status`

Description

Print detailed information about the current status of the device.

eth-hub-link

Provided By
[eth-links](#)

Interfaces Implemented
conf_object, [ethernet_snoop](#), [log_object](#)

Description
Simple broadcasting ethernet link

Attributes

goal_latency
Required attribute; **read/write** access; type: float. The desired latency for this link.

Command List

Commands
[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

<eth-hub-link>.info

Synopsis
<eth-hub-link>.info

Description
Print detailed information about the configuration of the device.

<eth-hub-link>.status

Synopsis
<eth-hub-link>.status

Description
Print detailed information about the current status of the device.

eth-hub-link-endpoint

Provided By
[eth-links](#)

Interfaces Implemented
conf_object, [ethernet_common](#), [log_object](#)

Description
Ethernet hub link endpoint

Attributes

device

Required attribute; **read/write** access; type: [os], object, or nil. The device connected to this endpoint.

id

Required attribute; **read/write** access; type: integer. Endpoint ID. The ID of each endpoint must be unique among the link's endpoints, and it may not be 0 or 0xfffffffffffffff.

link

Required attribute; **read/write** access; type: object. The link object to which this endpoint belongs.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<eth-hub-link-endpoint>.info

Synopsis

<eth-hub-link-endpoint>.info

Description

Print detailed information about the configuration of the device.

<eth-hub-link-endpoint>.status

Synopsis

<eth-hub-link-endpoint>.status

Description

Print detailed information about the current status of the device.

eth-link-snoop-endpoint

Provided By
[eth-links](#)

Interfaces Implemented
conf_object, [log_object](#)

Description
Ethernet link snoop endpoint

Attributes

id

Required attribute; **read/write** access; type: `integer`. Endpoint ID. The ID of each endpoint must be unique among the link's endpoints, and it may not be 0 or 0xfffffffffffffff

link

Required attribute; **read/write** access; type: `object`. The link object to which this endpoint belongs.

Command List

Commands

[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

`<eth-link-snoop-endpoint>.info`

Synopsis

`<eth-link-snoop-endpoint>.info`

Description

Print detailed information about the configuration of the device.

`<eth-link-snoop-endpoint>.status`

Synopsis

`<eth-link-snoop-endpoint>.status`

Description

Print detailed information about the current status of the device.

eth-probe

Provided By
[eth-probe](#)

Interfaces Implemented

conf_object, [ethernet_probe](#), [log_object](#)

Description

Ethernet probe. The probe can be inserted between any devices or link talking via the ethernet_common interface. A snooper or probe function can be attached to listen to the traffic going to the probe, and possibly modify it. Use command insert-ethernet-probe to create and insert a probe. Or do a manual connection instead by using command create-unconnected-ethernet-probe and assign the attributes in port A and B of the probe.

Attributes

probe_ports

Required attribute; **read/write** access; type: [oo]. Port objects of the probe as [Port A, Port B]

Command List

Commands

[delete](#)

[info](#)

print information about the device

[pcap-dump](#)

Dump network traffic to a file, in libpcap format

[pcap-dump-stop](#)

stop the current dump

[tcpdump](#)

run the tcpdump program

[tcpdump-stop](#)

stop the current tcpdump capture

[wireshark](#)

run the wireshark/ethereal program

[wireshark-stop](#)

stop the current wireshark capture

Command Descriptions

<eth-probe>.delete

Synopsis

<eth-probe>.delete

Description

<eth-probe>.info

Synopsis

<eth-probe>.info

Description

Print detailed information about the configuration of the device.

<eth-probe>.pcap-dump**Synopsis**

<eth-probe>.pcap-dump *file*
<eth-probe>.pcap-dump-stop

Description**<eth-probe>.pcap-dump-stop****Synopsis**

<eth-probe>.pcap-dump-stop
<eth-probe>.pcap-dump *file*

Description**<eth-probe>.tcpdump****Synopsis**

<eth-probe>.tcpdump [“*flags*”]
<eth-probe>.tcpdump-stop

Description

Runs the **tcpdump** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to **tcpdump**.

<eth-probe>.tcpdump-stop**Synopsis**

<eth-probe>.tcpdump-stop
<eth-probe>.tcpdump [“*flags*”]

Description

Runs the **tcpdump** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to **tcpdump**.

<eth-probe>.wireshark

Alias

<eth-probe>.ethereal

Synopsis

<eth-probe>.wireshark [“flags”]
<eth-probe>.wireshark-stop

Description

Runs the **wireshark** or **ethereal** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to program.

<eth-probe>.wireshark-stop

Alias

<eth-probe>.ethereal-stop

Synopsis

<eth-probe>.wireshark-stop
<eth-probe>.wireshark [“flags”]

Description

Runs the **wireshark** or **ethereal** program in a separate console, with network traffic captured from the simulated ethernet network. The *flags* are passed on unmodified to program.

eth-probe-port

Provided By
[eth-probe](#)

Interfaces Implemented
conf_object, [ethernet_common](#), [log_object](#)

Description
Ethernet probe port

Attributes

partner

Required attribute; **read/write** access; type: [os], object, or nil. Object to which the port is connected and talking Ethernet

partner_attr

Required attribute; **read/write** access; type: string. Attribute of the connected object that points at this probe port, if available

probe

Required attribute; **read/write** access; type: object. Probe object to which this port belongs

side

Required attribute; **read/write** access; type: integer. Whether this port is port A (0) or port B (1)

eth-switch-link

Provided By
[eth-links](#)

Interfaces Implemented

`conf_object, ethernet_snoop, ethernet_vlan_snoop, log_object`

Description

Switched ethernet link

Attributes

goal_latency

Required attribute; **read/write** access; type: `float`. The desired latency for this link.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<eth-switch-link>.info

Synopsis

`<eth-switch-link>.info`

Description

Print detailed information about the configuration of the device.

<eth-switch-link>.status

Synopsis

`<eth-switch-link>.status`

Description

Print detailed information about the current status of the device.

eth-switch-link-endpoint

Provided By
[eth-links](#)

Interfaces Implemented
conf_object, [ethernet_common](#), [log_object](#)

Description
Ethernet switch link endpoint

Attributes

device

Required attribute; **read/write** access; type: [os], object, or nil. The device connected to this endpoint.

id

Required attribute; **read/write** access; type: integer. Endpoint ID. The ID of each endpoint must be unique among the link's endpoints, and it may not be 0 or 0xffffffffffff

link

Required attribute; **read/write** access; type: object. The link object to which this endpoint belongs.

vlan_id

Required attribute; **read/write** access; type: integer or nil. The VLAN ID of the endpoint. If the endpoint is a trunk, the value is the native VLAN ID (1 - 4095) or None if the trunk should not have any native ID.

vlan_trunk

Required attribute; **read/write** access; type: boolean. Set to true if the endpoint is a trunk.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

`<eth-switch-link-endpoint>.info`

Synopsis

`<eth-switch-link-endpoint>.info`

Description

Print detailed information about the configuration of the device.

```
<eth-switch-link-endpoint>.status
```

Synopsis

```
<eth-switch-link-endpoint>.status
```

Description

Print detailed information about the current status of the device.

eth-switch-link-snoop-endpoint

Provided By
[eth-links](#)

Interfaces Implemented
conf_object, [log_object](#)

Description
Ethernet switch snoop endpoint

Attributes

id

Required attribute; **read/write** access; type: `integer`. Endpoint ID. The ID of each endpoint must be unique among the link's endpoints, and it may not be 0 or 0xfffffffffffffff

link

Required attribute; **read/write** access; type: `object`. The link object to which this endpoint belongs.

vlan_id

Required attribute; **read/write** access; type: `integer` or `nil`. The VLAN ID of the endpoint. If the endpoint is a trunk, the value is the native VLAN ID (1 - 4095) or None if the trunk should not have any native ID.

vlan_trunk

Required attribute; **read/write** access; type: `boolean`. Set to true if the endpoint is a trunk.

Command List

Commands

[info](#) print information about the device
[status](#) print status of the device

Command Descriptions

`<eth-switch-link-snoop-endpoint>.info`

Synopsis

`<eth-switch-link-snoop-endpoint>.info`

Description

Print detailed information about the configuration of the device.

`<eth-switch-link-snoop-endpoint>.status`

Synopsis

<eth-switch-link-snoop-endpoint>.status

Description

Print detailed information about the current status of the device.

eth-transceiver

Provided By
[eth-transceiver](#)

Interfaces Implemented
[conf_object](#), [log_object](#), [mii](#), [mii_management](#)

Ports
MDC ([signal](#)), MDIO ([signal](#))

Description
eth-transceiver is an IEEE 802.3 physical layer device with MII interface

Attributes

phyidr1
Required attribute; **read/write** access; type: `integer`. Value of the PHYIDR1 register.

phyidr2
Required attribute; **read/write** access; type: `integer`. Value of the PHYIDR2 register.

Command List

Commands

connect	<i>deprecated</i> — connect to a simulated Ethernet link
disconnect	<i>deprecated</i> — disconnect from simulated link
info	print information about the device
status	print status of the device

Command Descriptions

<eth-transceiver>.connect — *deprecated*

Synopsis
`<eth-transceiver>.connect [-auto] link`

Description
This command is deprecated; use [connect](#) instead.

Connect the device to a simulated Ethernet link. The flag '-auto' is deprecated and shouldn't be used.

See Also

[<eth-transceiver>.disconnect](#)

<eth-transceiver>.disconnect — *deprecated*

Synopsis

<eth-transceiver>.disconnect

Description

This command is deprecated; use [disconnect](#) instead.

Disconnect the device from a simulated Ethernet link.

See Also

[<eth-transceiver>.connect](#)

<eth-transceiver>.info

Synopsis

<eth-transceiver>.info

Description

Print detailed information about the configuration of the device.

<eth-transceiver>.status

Synopsis

<eth-transceiver>.status

Description

Print detailed information about the current status of the device.

eth_injector

Provided By
[eth-injector](#)

Interfaces Implemented
conf_object, [ethernet_common](#), [log_object](#)

Description

Peudo ethernet device for packet injection based on pcap file input.

Command List

Commands

- info** print information about the device
- start** Start pcap playback
- status** print status of the device

Command Descriptions

<eth_injector>.info

Synopsis

<eth_injector>.info

Description

Print detailed information about the configuration of the device.

<eth_injector>.start

Synopsis

<eth_injector>.start *file* [-no-crc]

Description

Start injecting files from the pcap file *file*. If packets in the pcap file has no Ethernet CRC included use the *-no-crc* flag. This will append a dummy CRC to the frame and the simulation handle the packet as if the CRC is correct. If the *-no-crc* was not given, the frame will be injected without modification.

<eth_injector>.status

Synopsis

<eth_injector>.status

Description

Print detailed information about the current status of the device.

ethernet-link

Provided By
[ethernet-link](#)

Interfaces Implemented
conf_object, [ethernet_link](#), [log_object](#)

Description
obsolete Ethernet link

Attributes

central

Optional attribute; **read/write** access; type: object or nil. The Simics Central client object used for distributing the simulation. (or a cell)

devices

Optional attribute; **read/write** access; type: `[[i|n, i, s, o|n, [[[ss] *], b]] *]`. The connected devices. It is a list of entries, where each entry is a list [*global id, local id, name, obj, info*]. The *obj* is either a reference to the device object for local devices, or nil if the device was in another Simics process. The *info* field is a list [*macs, promisc*], where *macs* is a list of pairs [*addr, mask*].

filter_enable

Optional attribute; **read/write** access; type: boolean. A flag controlling whether the MAC information is used to only deliver frames to devices that will handle them. A change in this flag might change the result of the simulation in some corner cases, but is usually transparent. If it is enabled, it may help simulation efficiency.

frame_echo

Optional attribute; **read/write** access; type: boolean. This flag defines if the link will echo frames back to the sender. Default is not to echo frames.

frequency

Optional attribute; **read/write** access; type: integer. The frequency of the clock used by the link to time stamp traffic and state changes. This defines the granularity of the time of any event on the link. The frequency is given as number of ticks per second.

last_frame

Pseudo attribute; **read/write** access; type: data or nil. This attribute contains the next frame to deliver, when read from an Ethernet_Frame hap handler. It can also be set from the hap handler, which will modify the packet being sent. If it is set to nil or to an empty buffer, the packet will be dropped. From anywhere else than an Ethernet_Frame hap handler, this attribute is nil and can't be set.

latency

Optional attribute; **read/write** access; type: `float` or `integer`. The latency on the link. Every packet that is sent over this link is delayed by at least the time specified in this attribute. Other simulated state changes, such as auto-negotiation and addresses also propagate over the link with the same delay. The type of this attribute is either an integer number of clock ticks in the time base specified in the *frequency* attribute, or a floating point number which defines the latency in seconds.

link_id

Pseudo attribute; **read/write** access; type: `integer`. The ID number of the link this link object belongs to.

link_obj_id

Pseudo attribute; **read/write** access; type: `integer`. The local ID number of this link object on the link.

linkname

Optional attribute; **read/write** access; type: `string` or `nil`. The global name of the link that this link objects belongs to. This name is used to identify links that are shared by several simulations. If this attribute is `nil`, the link can only be used locally.

master_obj_id

Pseudo attribute; **read/write** access; type: `integer`. The local ID number of the master link object on the link.

min_latency

Pseudo attribute; **read/write** access; type: `integer`. The minimum latency allowed. This is an integer number of clock ticks in the time base specified in the *frequency* attribute. In a distributed the value is given by Simics Central server. This attribute can not be changed.

nbytes

Optional attribute; **read/write** access; type: `integer`. The number of bytes transmitted over the network. This does not include any frames still in transfer.

next_seq

Optional attribute; **read/write** access; type: `integer`. The sequence number assigned to the next frame sent over the link

nframes

Optional attribute; **read/write** access; type: `integer`. The number of frames transmitted over the network. This does not include any frames still in transfer.

pending_frames

Optional attribute; **read/write** access; type: `[[ibi|[o[ii]|n]ibd]*]`. Frames in transfer

promisc_default

Optional attribute; **read/write** access; type: boolean. This attribute is internal and may go away in the future

vlan

Optional attribute; **read/write** access; type: dictionary or nil. The VLAN configuration for the pseudo-switch in the ethernet link. It is a dictionary that maps device objects to a VLAN ID (VID), which will make that device part of that VLAN whenever it is connected. The VLAN ID is a 12-bit integer or the special string value “trunk”, which means that the object should receive all traffic, tagged with 802.1Q tags, and that incoming traffic should be untagged and sent to the appropriate VLAN.

A connected device that is not mentioned in this attribute is considered to have VID 1.

If this attribute is nil, VLAN tags are ignored and all packets are passed on unmodified.

Command List

Commands

connect-real-network-bridge	connect to the real network
connect-real-network-host	connect to the real network
connect-real-network-napt	enable NAPT from simulated network
connect-real-network-router	<i>deprecated</i> — connect to the real network
disconnect-real-network	disconnect from the real network
info	print information about the device
status	print status of the device

Command Descriptions

<ethernet-link>.connect-real-network-bridge

Synopsis

```
<ethernet-link>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
<ethernet_cable>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
<ethernet_hub>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
<ethernet_switch>.connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
connect-real-network-bridge ["interface"] ["host-access"] [-no-mac-xlate] [-persistent] [-propagate-link-status]
```

Description

Creates an Ethernet bridge between a simulated Ethernet link and a real network through an Ethernet interface of the simulation host.

The optional *interface* argument specifies the Ethernet or TAP interface of the host to use.

By default a TAP interface is used, but if the *host-access* argument is `raw`, raw access to an Ethernet interface is used.

MAC address translation can be disabled with the `-no-mac-xlate` flag.

The `-persistent` is for backward compatibility and should not be used.

If `-propagate-link-status` is specified, link status changes on the host interface will be propagated to all devices on the link that implements the link-status interface. For TAP, only 'up' and 'down' status will be propagated (and not 'unconnected'). Link status propagation is only supported on Linux.

This command returns the new real-network component object.

See the *Connecting to a Real Network* chapter of the *Ethernet Networks in Simics* manual for more information about how to connect to a real network.

See Also

[connect-real-network](#), [connect-real-network-host](#), [disconnect-real-network](#)

`<ethernet-link>.connect-real-network-host`

Synopsis

```
<ethernet-link>.connect-real-network-host ["interface"] [-persistent]
<ethernet_cable>.connect-real-network-host ["interface"] [-persistent]
<ethernet_hub>.connect-real-network-host ["interface"] [-persistent]
<ethernet_switch>.connect-real-network-host ["interface"] [-persistent]
connect-real-network-host ["interface"] [-persistent]
```

Description

Connects a TAP interface of the simulation host to a simulated Ethernet link.

The optional *interface* argument specifies the TAP interface of the host to use.

The `-persistent` is for backward compatibility and should not be used.

This command returns the new real-network component object.

See the *Connecting to a Real Network* chapter of the *Simics User Guide* for more information about how to connect to a real network.

See Also

[connect-real-network](#), [connect-real-network-bridge](#), [disconnect-real-network](#)

`<ethernet-link>.connect-real-network-napt`

Synopsis

```
<ethernet-link>.connect-real-network-napt [service-node]
<ethernet_cable>.connect-real-network-napt [service-node]
<ethernet_hub>.connect-real-network-napt [service-node]
<ethernet_switch>.connect-real-network-napt [service-node]
<ethernet_vlan_switch>.connect-real-network-napt [service-node]
connect-real-network-napt ethernet-link [service-node]
```

Description

Enables machines on the simulated network to initiate accesses to real hosts without the need to configure the simulated machine with a real IP address. NAPT (Network Address Port Translation) uses the IP address and a port number of the host that Simics is running on to perform the access. Replies are then translated back to match the request from the simulated machine. This command also enables NAPT for accesses that are initiated from the simulated machine.

See Also

[connect-real-network](#), [connect-real-network-port-in](#), [connect-real-network-port-out](#),
[connect-real-network-host](#), [connect-real-network-bridge](#)

`<ethernet-link>.connect-real-network-router — deprecated`

Synopsis

```
<ethernet-link>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]

<ethernet_cable>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]

<ethernet_hub>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]

<ethernet_switch>.connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]

connect-real-network-router "ip" ["netmask"] ["gateway"] ["interface"]
```

Description

This command is deprecated; use [connect-real-network-bridge](#) or [connect-real-network-host](#) instead.

Deprecated command

See Also

[connect-real-network](#), [connect-real-network-host](#), [connect-real-network-bridge](#), [disconnect-real-network](#)

`<ethernet-link>.disconnect-real-network`

Synopsis

```
<ethernet-link>.disconnect-real-network
<ethernet_cable>.disconnect-real-network
<ethernet_hub>.disconnect-real-network
<ethernet_switch>.disconnect-real-network
disconnect-real-network
```

Description

Closes all connections to real networks except port forwarding and NAPT.

See Also

[connect-real-network-host](#), [connect-real-network-bridge](#)

<ethernet-link>.info

Synopsis

<ethernet-link>.info

Description

Print detailed information about the configuration of the device.

<ethernet-link>.status

Synopsis

<ethernet-link>.status

Description

Print detailed information about the current status of the device.

fake-space

Provided By

[Simics Core](#)

Interfaces Implemented

[breakpoint](#), [conf_object](#), [io_memory](#), [log_object](#), [memory_space](#)

Description

Translates all memory-space interface methods that are not exposed to Python, to methods that are. This is intended for module testing.

Attributes

target_space

Required attribute; **read/write** access; type: `object`. Memory space to which all interface access will be redirected. Usually implemented in Python or similar.

Command List

Commands defined by interface [breakpoint](#)

[break](#), [tbreak](#)

Commands defined by interface [memory_space](#)

[debug](#)

fc-disk

Provided By

[fc-disk](#)

Interfaces Implemented

[conf_object](#), [log_object](#)

Description

Simple model of a Fibre Channel SCSI disk for use with the ISP2200 controller. This device and the controller do not actually use the FC protocol since it is hidden by the controller from the host.

Attributes

fc_controller

Required attribute; **read/write** access; type: `object`. Name of the Fibre Channel controller that this disk is connected to.

geometry

Required attribute; **read/write** access; type: `[iii]`. The geometry of the disk. ()

image

Required attribute; **read/write** access; type: `object`. Name of the image object holding the actual data of the disk. This object must implement the 'image' interface.

loop_id

Required attribute; **read/write** access; type: `integer`. Loop ID on the FC-AL.

node_name

Required attribute; **read/write** access; type: `integer`. Unique name of the node, as 48 bit IEEE address

port_name

Required attribute; **read/write** access; type: `integer`. Unique name of the port, as 48 bit IEEE address

Command List

Commands

[add-diff-file](#)

add a diff file to the image

[add-diff-partial-file](#)

add a partial diff file to the image

[add-sun-partition](#)

add partition from a file

[create-sun-vtoc-header](#)

write a new VTOC to a Sun disk

[create-sun-vtoc-partition](#)

write partition data in the VTOC on a Sun disk

[delete-sun-vtoc-partition](#)

delete partition data from the VTOC on a Sun disk

[dump-sun-partition](#)

write partition as a file

[info](#)

information about current state of the fibre-channel disk

[print-sun-vtoc](#)

print the VTOC for a Sun disk

[save-diff-file](#)

save diff file to disk

Command Descriptions

`<fc-disk>.add-diff-file`

Synopsis

`<fc-disk>.add-diff-file filename [-replace] [-rw]`

Description

Add a diff file to the list of files for an disk. The diff file was typically created with the 'save-diff-file' command, or by a saved configuration. This is basically the same command as `<image>.add-diff-file`. The file can be made writable instead of read-only using the `-rw` flag. To replace any existing files, use `-replace`.

See Also

[`<fc-disk>.save-diff-file`](#), [`<image>.add-diff-file`](#)

`<fc-disk>.add-diff-partial-file`

Synopsis

`<fc-disk>.add-diff-partial-file filename start [size]`

Description

Add a diff file to the list of files for an disk. The diff file was typically created with the 'save-diff-file' command, by one of the `dump-*-partition` commands, or by a saved configuration. This is basically the same command as `<image>.add-partial-diff-file`.

See Also

[`<fc-disk>.save-diff-file`](#), [`<image>.add-diff-file`](#), [`<image>.add-partial-diff-file`](#)

`<fc-disk>.add-sun-partition`

Synopsis

`<fc-disk>.add-sun-partition number file`

Description

Adds an image or diff as a sun partition to the current disk.

See Also

[`<fc-disk>.dump-sun-partition`](#)

`<fc-disk>.create-sun-vtoc-header`

Synopsis

`<fc-disk>.create-sun-vtoc-header [C] [H] [S] [-quiet]`

Description

Create and write a new VTOC to a Sun disk. The geometry information written is taken from the configuration attribute 'geometry' of the disk, unless specified with the **C**, **H** and **S** parameters. A new empty partition table is also created, with only the standard 'backup' partition as number 2. `-quiet` makes the command silent in case of success.

See Also

[`<fc-disk>.print-sun-vtoc`](#), [`<fc-disk>.create-sun-vtoc-partition`](#), [`<fc-disk>.delete-sun-vtoc-partition`](#)

`<fc-disk>.create-sun-vtoc-partition`

Synopsis

`<fc-disk>.create-sun-vtoc-partition` *number* "tag" "flag" *start-block num-blocks [-quiet]*

Description

Write partition information to the VTOC on a Sun disk. This command does not change the format of the disk, and it does not create any file system on the partition. Only the 'Volume Table Of Contents' is modified. No checking is performed to make sure that partitions do not overlap, or that they do not exceed the disk size. *-quiet* makes the command silent in case of success.

See Also

[`<fc-disk>.print-sun-vtoc`](#), [`<fc-disk>.create-sun-vtoc-header`](#), [`<fc-disk>.delete-sun-vtoc-partition`](#)

`<fc-disk>.delete-sun-vtoc-partition`

Synopsis

`<fc-disk>.delete-sun-vtoc-partition` *number* [*-quiet*]

Description

Delete the information in the VTOC on a Sun disk for the specified partition. No other modification on the disk is performed. *-quiet* makes the command silent in case of success.

See Also

[`<fc-disk>.print-sun-vtoc`](#), [`<fc-disk>.create-sun-vtoc-header`](#), [`<fc-disk>.create-sun-vtoc-partition`](#)

`<fc-disk>.dump-sun-partition`

Synopsis

`<fc-disk>.dump-sun-partition` *number file*

Description

Write all data from a Sun disk partition to the specified file in raw format.

See Also

[`<fc-disk>.print-sun-vtoc`](#), [`<fc-disk>.add-sun-partition`](#)

`<fc-disk>.info`

Synopsis

`<fc-disk>.info`

Description

Print information about the state of the fibre-channel disk.

`<fc-disk>.print-sun-vtoc`

Synopsis

`<fc-disk>.print-sun-vtoc`

Description

Print the contents of the VTOC (volume table of contents) for a Sun disk. This is similar to the Solaris 'prtvtoc' command.

See Also

[`<fc-disk>.create-sun-vtoc-header`](#), [`<fc-disk>.create-sun-vtoc-partition`](#), [`<fc-disk>.delete-sun-vtoc-partition`](#)

`<fc-disk>.save-diff-file`

Synopsis

`<fc-disk>.save-diff-file filename`

Description

Writes changes to the image as a diff file in craff format. This is basically the same command as `<image>.save-diff-file`.

See Also

[`<fc-disk>.add-diff-file`](#), [`<image>.save-diff-file`](#)

file-cdrom

Provided By
[file-cdrom](#)

Interfaces Implemented
`conf_object`, [log_object](#)

Description

This class allows a file in ISO9660 format to be accessible from the simulated target machine. This is a convenient way to import files into the simulated system.

Attributes

file

Required attribute; **read/write** access; type: `string`. The name of the ISO file to use as CD-ROM media.

Command List

Commands

[**delete**](#) delete an unused file-cdrom object

Command Descriptions

`<file-cdrom>.delete`

Synopsis

`<file-cdrom>.delete`

Description

Delete an unused file-cdrom object with the name *object-name*.

firewire_bus

Provided By
[firewire-bus](#)

Interfaces Implemented
`conf_object`, [firewire_bus](#), [io_memory](#), [log_object](#)

Description

A FireWire bus. The bus takes responsibility for all the bus arbitration and assigning ids. It does not keep track of the topology of the bus, so the IDs will be quite arbitrary. Neither does it keep track of the transfer rate limitations, which means that you can transmit as much data per time unit as you want.

Attributes

connections

Optional attribute; **read/write** access; type: `[[[oi][oi]]*]`. The connections between the ports on the devices. Each connection is represented as a pair of ports, where each port is represented as a pair of a device and a port number. The connections must be kept consistent with the devices connected to the bus.

current_transfer

Pseudo attribute; **read/write** access; type: `data` or `nil`. The current transfer. This attribute is only valid during the `Firewire_Transfer` hap.

self_ids

Pseudo attribute; **read/write** access; type: `[i*]`. The self-id packets describing the current tree. This attribute is only valid during the `Firewire_Bus_Reset` hap. It describes the structure of the tree. The `connected_devices` attribute is what is used to say which device has which node-id.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

`<firewire_bus>.info`

Synopsis

`<firewire_bus>.info`

Description

Print detailed information about the configuration of the device.

<firewire_bus>.status

Synopsis

<firewire_bus>.status

Description

Print detailed information about the current status of the device.

firewire_device_test_wrapper

Provided By

[firewire-bus](#)

Interfaces Implemented

`conf_object`, [firewire_device](#), [log_object](#)

Description

Simple device used for tests

Attributes

packet_data_after_transfer

Pseudo attribute; **read-only** access; type: `data` or `nil`. Content of the last packet after `real_device.transfer` was called.

packet_data_before_transfer

Pseudo attribute; **read-only** access; type: `data` or `nil`. Content of the last packet before `real_device.transfer` was called.

real_device

Required attribute; **read/write** access; type: `[os]` or `object`. The real firewire device the calls are forwarded to

firewire_sample_device

Provided By

[firewire-sample-device](#)

Interfaces Implemented

[conf_object](#), [firewire_device](#), [log_object](#)

Ports

PHY ([int_register](#), [io_memory](#)), config_rom ([int_register](#), [io_memory](#)), firewire_config_registers ([int_register](#), [io_memory](#))

Description

A simple firewire device which only implements the required firewire registers. It can't take on the cycle-master, isochronous resource manager or bus manager roles.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<firewire_sample_device>.info

Synopsis

<firewire_sample_device>.info

Description

Print detailed information about the configuration of the device.

<firewire_sample_device>.status

Synopsis

<firewire_sample_device>.status

Description

Print detailed information about the current status of the device.

flash-memory

Provided By
[flash-memory](#)

Interfaces Implemented
[conf_object](#), [io_memory](#), [log_object](#), [translate](#)

Description

This model is deprecated. If you want to create new flash-memory objects, use generic-flash-memory instead.

Attributes

storage_ram

Required attribute; **read/write** access; type: `object`. RAM object providing the backing store area.

frequency_bus

Provided By
[frequency-bus](#)

Interfaces Implemented

`conf_object`, [frequency_listener](#), `log_object`, `scale_factor_listener`, [simple_dispatcher](#)

Description

The `frequency_bus` class implements a bus that distributes a frequency to a set of targets. Targets subscribe to the bus via the `simple_dispatcher` interface, and frequency changes are distributed to subscribers via the `frequency_listener` interface. The distributed frequency can either be generated by the bus itself, or be fetched from a different object (e.g., from another frequency bus). The input frequency can optionally be scaled before it is distributed.

Attributes

current_output_freq

Pseudo attribute; **read-only** access; type: `[ii]`. The frequency currently output to connected devices

frequency

Optional attribute; **read/write** access; type: `[ii]`, `[os]`, or `object`. The frequency broadcasted by this bus. A value of `[p, q]` means a fix frequency of p/q Hz; if the value is an object implementing the `simple_dispatcher` interface, the frequency is fetched from that object; if the value is an `[object, port]` pair, the frequency is fetched from the given port.

scale_factor

Optional attribute; **read/write** access; type: `[ii]`, `[os]`, or `object`. The input frequency is multiplied by this number before broadcasted. A value of `[p, q]` means a fix factor of p/q ; if the value is an object implementing the `simple_dispatcher` interface, the factor is fetched from that object; if the value is an `[object, port]` pair, the factor is fetched from the given port.

targets

Pseudo attribute; **read-only** access; type: `[[o, s|n]*]`. Current frequency target objects, on the form `[object, port]`

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<frequency_bus>.info

Synopsis

<frequency_bus>.info

Description

Print detailed information about the configuration of the device.

<frequency_bus>.status

Synopsis

<frequency_bus>.status

Description

Print detailed information about the current status of the device.

frontend-server

Provided By
[frontend-server](#)

Interfaces Implemented
[attribute_monitor](#), [conf_object](#), [log_object](#), [remote_callback](#)

Description
Module used for communicating with a frontend client

frontend-server-console

Provided By
[frontend-server-console](#)

Interfaces Implemented
conf_object, [log_object](#)

Description
frontend-server console proxy

ftp-alg

Provided By

[ftp_alg](#)

Interfaces Implemented

`conf_object`, [log_object](#)

Description

FTP ALG.

Attributes

forward_handler

Required attribute; **read/write** access; type: `object`. An instance of the port-forward-outgoing-server class. Must implement the `port_forward` interface.

incoming_handler

Required attribute; **read/write** access; type: `object`. Must implement the `port_forward` interface.

ftp-control

Provided By

[ftp-service](#)

Interfaces Implemented

[conf_object](#), [log_object](#)

Description

Helper class for the ftp-server. Used to keep track of control connections

Attributes

ftp

Required attribute; **read/write** access; type: `object`. FTP object

Command List

Commands

[info](#) print information about the device

[status](#) print status of the device

Command Descriptions

<ftp-control>.info

Synopsis

`<ftp-control>.info`

Description

Print detailed information about the configuration of the device.

<ftp-control>.status

Synopsis

`<ftp-control>.status`

Description

Print detailed information about the current status of the device.

ftp-data

Provided By

[ftp-service](#)

Interfaces Implemented

[conf_object](#), [log_object](#)

Description

Helper class for the ftp-server. Used to keep track of data connections

Attributes

ftp

Required attribute; **read/write** access; type: `object`. FTP object

Command List

Commands

[**info**](#) print information about the device

[**status**](#) print status of the device

Command Descriptions

<ftp-data>.info

Synopsis

`<ftp-data>.info`

Description

Print detailed information about the configuration of the device.

<ftp-data>.status

Synopsis

`<ftp-data>.status`

Description

Print detailed information about the current status of the device.

ftp-service

Provided By

[ftp-service](#)

Interfaces Implemented

`conf_object`, [log_object](#)

Description

The ftp-service class implements a simple FTP server in the service node. There is no authentication, all user names and passwords will be accepted and give access to all files the Simics process may access. The most commonly used FTP commands are implemented such as RETR, STOR, PWD, CWD, CDUP, MODE, USER, PASS, TYPE, PASV, PORT, NOOP, STRU, LIST, NSLT and QUIT. Unimplemented features include IPv6, creation and deletion of directories and file deletion and renaming. ASCII transfers will handled as binary ones.

Attributes

c_sessions

Optional attribute; **read/write** access; type: `[[ssisi] *]`. The active control sessions in the FTP service. The sub-entries for each session are: *server IP address*, *client IP address*, *client port*, *current directory*, *dsession id*.

d_sessions

Optional attribute; **read/write** access; type: `[[bbbs|niisi] *]`. The active data sessions in the FTP service. The sub-entries for each session are: *passive mode*, *connected (in passive mode)*, *write file name*, *file offset*, *write buffer size*, *directory listing*, *c_session id*.

enabled

Optional attribute; **read/write** access; type: `boolean`. Set to true (default) if the FTP server will listen for connections on the IP addresses specified in the *server_ip_list* attribute.

ftp_helpers

Required attribute; **read/write** access; type: `[oo]`. FTP helper objects. One ftp-control object and one ftp-data object.

server_ip_list

Required attribute; **read/write** access; type: `[s*]`. Server IPv4 addresses. Typically one for each interface of the service node. New addresses may be added at the end of the list, but removal is not supported.

tcp

Required attribute; **read/write** access; type: `object`. TCP layer. Must implement the *tcp* interface.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<ftp-service>.info

Synopsis

<ftp-service>.info

Description

Print detailed information about the configuration of the device.

<ftp-service>.status

Synopsis

<ftp-service>.status

Description

Print detailed information about the current status of the device.

g-cache

Provided By
[g-cache](#)

Interfaces Implemented
conf_object, [log_object](#), [timing_model](#)

Ports

c_dev_data_read (scalar_time), c_dev_data_write (scalar_time), copy_back (scalar_time), data_read (scalar_time), data_read_miss (scalar_time), data_write (scalar_time), data_write_miss (scalar_time), dev_data_read (scalar_time), dev_data_write (scalar_time), inst_fetch (scalar_time), inst_fetch_miss (scalar_time), mesi_exclusive_to_shared (scalar_time), mesi_invalidate (scalar_time), mesi_modified_to_shared (scalar_time), transaction (scalar_time), uc_data_read (scalar_time), uc_data_write (scalar_time), uc_inst_fetch (scalar_time)

Description

g-cache is an in-order 'flat' cache model with the following features: configurable number of lines, line size, associativity; indexing/tagging on physical/virtual addresses; configurable policies for write-allocate and write-back; random, cyclic and lru replacement policies; several caches can be connected in chain; a single cache can be connected to several processors; support for a simple MESI protocol between caches.

Attributes

cpus

Required attribute; **read/write** access; type: [o*], object, or nil. cpus that can send transactions to the cache.

Command List

Commands

add-profiler	Add a profiler to the cache
info	print the cache information
remove-profiler	Remove a profiler from the cache
reset-cache-lines	Reset all the cache lines
reset-statistics	reset the cache statistics
statistics	print the cache statistics
status	print the cache lines status

Command Descriptions

<g-cache>.add-profiler

Synopsis

<g-cache>.add-profiler ["*type*"] ["*profiler*"]

Description

Add a profiler of the given 'type' to the cache. If a 'profiler' is passed as argument, it will be used instead of a newly created object.

<g-cache>.info**Synopsis**

<g-cache>.info

Description

Print configuration information for the cache.

<g-cache>.remove-profiler**Synopsis**

<g-cache>.remove-profiler ["*type*"]

Description

Remove the profiler of a given 'type' from the cache.

<g-cache>.reset-cache-lines**Synopsis**

<g-cache>.reset-cache-lines

Description

Reset all the cache lines to their default values.

<g-cache>.reset-statistics**Synopsis**

<g-cache>.reset-statistics

Description

Reset the cache statistics.

<g-cache>.statistics**Synopsis**

<g-cache>.statistics

Description

Print the current value of the cache statistics (this will flush the STC counters to report correct values).

<g-cache>.status

Synopsis

<g-cache>.status [-cycle]

Description

Print the cache lines status. -cycle prints the last cycle each cache line was accessed (if LRU replacement policy is active). A '<' marks the next line to be replaced (if cyclic replacement policy is active).

gdb-remote

Provided By

[gdb-remote](#)

Interfaces Implemented

`conf_object`, [log_object](#)

Description

The **gdb-remote** module allows a GDB session to connect to Simics and control the execution. An object of class **gdb-remote** is used to accept incoming GDB connection requests.

A gdb binary capable of debugging any Simics target machine is included in the Simics Base package. If you want to build your own gdb, read on.

The following table lists, for each target architectures supported by **gdb-remote**, the string to give to `configure` as the `--target` parameter when building GDB, and any command you may have to enter at the GDB command prompt before connecting to Simics:

arc600

```
--target arc-elf32  
command: set architecture ARC600
```

arc700

```
--target arc-elf32  
command: set architecture ARC700
```

armbe

```
--target armbe-unknown-linux-gnu
```

armle

```
--target armle-unknown-linux-gnu
```

h8300

```
--target h8300-elf  
command: set architecture h8300(s|h)
```

m68k

```
--target m68k  
command: set architecture m68k
```

mips32be

```
--target mips-elf-linux  
command: set architecture mips:isa32r2
```

mips32le

```
--target mips-elf-linux  
command: set architecture mips:isa32r2
```

```

mips64be
  --target mips64-elf64-linux64
  command: set architecture mips:isa64r2

mips64le
  --target mips64el-elf64-linux64
  command: set architecture mips:isa64r2

ppc32
  --target powerpc64-elf-linux
  command: set architecture powerpc:common

ppc64
  --target powerpc64-elf-linux
  command: set architecture powerpc:common64

ppce500
  --target powerpc64-elf-linux
  command: set architecture powerpc:e500

sparc-v8
  --target sparc-unknown-linux-gnu
  command: set architecture sparc:v8plus

sparc-v9
  --target sparc64-sun-solaris2.8
  command: set architecture sparc:v9

spu
  --target spu
  command: set architecture spu:256K

x86-64
  --target x86_64-pc-linux-gnu
  command: set architecture i386:x86-64

x86
  --target x86_64-pc-linux-gnu
  command: set architecture i386

```

Note that these --target flags are not the only ones that will work, just examples of ones that do work.

Command List

Commands

disconnect	disconnect from the remote gdb
follow-context	follow context
info	print information about the device
signal	tell remote gdb we got a signal
target	set target CPU for gdb connection

Command Descriptions

<gdb-remote>.disconnect

Synopsis

<gdb-remote>.disconnect

Description

Disconnect from the remote GDB. See the *Using Simics for Software Development* manual for a longer description of gdb-remote.

<gdb-remote>.follow-context

Synopsis

<gdb-remote>.follow-context [context]

Description

Set the GDB session to follow *context*. If *context* is not specified, the GDB session will stop following any context.

<gdb-remote>.info

Synopsis

<gdb-remote>.info

Description

Print detailed information about the configuration of the device.

<gdb-remote>.signal

Synopsis

<gdb-remote>.signal signal

Description

Send a signal to the remote GDB. See the *Using Simics for Software Development* manual for a longer description of gdb-remote.

<gdb-remote>.target

Synopsis

<gdb-remote>.target [cpu-name]

Description

Set the target CPU for this remote GDB connection. One GDB connection can only debug instructions on a single CPU at a time.

generic-flash-memory

Provided By

[new-flash-memory](#)

Aliases

new-flash-memory

Interfaces Implemented

conf_object, [log_object](#), translate

Ports

Reset ([signal](#)), wren ([signal](#))

Description

The generic-flash-memory class simulates different types of flash-memory depending on which attributes are set. Refer to [simics]/src/extensions/apps-python/flash_memory.py for a complete description of the features implemented and the flash chips that are pre-configured.

Limitations

- Many vendor-specific commands are not implemented.

Attributes

accept_smaller_reads

Pseudo attribute; **read/write** access; type: `integer`. Obsolete, do not use.

accept_smaller_writes

Pseudo attribute; **read/write** access; type: `integer`. Obsolete, do not use.

amd_ignore_cmd_address

Optional attribute; **read/write** access; type: `integer`. If 1, the address will be ignored when parsing AMD commands. Default is 0.

big_endian

Optional attribute; **read/write** access; type: `integer`. If 1, the flash device will behave as a big endian device. If 0, it will behave as a little endian device. Default is 0.

bus_width

Required attribute; **read/write** access; type: `integer`. Total width (in bits) of the data path connected to the flash device.

busy_signal_targets

Optional attribute; **read/write** access; type: `[n|o|[os]*]`. (`dst_object, dst_signal`)* The destination device and signal name to connect the busy signal of the chips to. The destinations should implement the `signal` interface. Without a timing model, the device will never raise the busy signal.

cfi_query

Optional attribute; **read/write** access; type: [i+], data, or nil. CFI query structure (if the device is CFI compatible). Default is none (device is not CFI compatible).

command_set

Optional attribute; **read/write** access; type: integer. If no CFI structure is provided, this attribute should be set to indicate the command-set to use. Default is 0 (invalid command-set).

device_id

Optional attribute; **read/write** access; type: [i+] or integer. Device ID/code as used in Intel identifier codes and AMD autoselect mode. Default is 0.

intel_configuration

Optional attribute; **read/write** access; type: integer. If 1, the flash device supports Intel configuration operations. If 0, Intel configuration operations are ignored. Default is 0.

intel_lock

Optional attribute; **read/write** access; type: integer. If 2, the flash device supports advanced lock/unlock/lock down operations. If 1, the flash device supports simple lock/unlock all operations. If 0, lock operations are ignored. Default is 0.

intel_protection_program

Optional attribute; **read/write** access; type: integer. If 1, the flash device supports Intel protection program operations. If 0, Intel protection program operations are ignored. Default is 0.

intel_write_buffer

Optional attribute; **read/write** access; type: integer. If 1, the flash device supports Intel write buffer operations. If 0, Intel write buffer operations are ignored. Default is 0.

interleave

Required attribute; **read/write** access; type: integer. Interleave (number of parallel flash memory chips).

manufacturer_id

Optional attribute; **read/write** access; type: integer. Manufacturer ID/code as used in Intel identifier codes and AMD autoselect mode. Default is 0.

max_chip_width

Optional attribute; **read/write** access; type: integer. Maximum data width (for example, specified as 16 for a x8/x16 capable device).

reset

Pseudo attribute; **write-only** access; type: integer. Set to 1 in order to reset the device.

storage_ram

Required attribute; **read/write** access; type: `object`. RAM object providing the backing store area.

strict_cmd_set

Optional attribute; **read/write** access; type: `integer`. If set to 1, warnings that the command-set is misused become errors. Default is 0.

timing_model

Optional attribute; **read/write** access; type: `dictionary; string indexed; indexed type: float`. Associates a flash state/operation with a time. The flash will remain in this state the given time allowing a more strict time model to be simulated. Sometimes flash drivers require that an operation takes some time for the software to work correctly.

unit_size

Required attribute; **read/write** access; type: `[i+]`. A list of block/sector sizes.

write_buffer_size

Optional attribute; **read/write** access; type: `integer`. Write buffer size *in bytes* for write buffer commands. Default is 32 (standard value for Intel Strataflash®).

Command List

Commands

accept-inquiries	<i>deprecated</i> — set whether or not to handle inquiry accesses
info	print information about the device
status	print status of the device

Command Descriptions

`<generic-flash-memory>.accept-inquiries — deprecated`

Synopsis

`<generic-flash-memory>.accept-inquiries [-on|-off]`

Description

This command is deprecated; use `<memory-space>.write` instead.

`<generic-flash-memory>.info`

Synopsis

`<generic-flash-memory>.info`

Description

Print detailed information about the configuration of the device.

<generic-flash-memory>.status

Synopsis

<generic-flash-memory>.status

Description

Print detailed information about the current status of the device.

generic-message-link

Provided By
[g-link](#)

Interfaces Implemented

[conf_object](#), [generic_message_link](#), [log_object](#)

Description

obsolete generic message link

Attributes

central

Optional attribute; **read/write** access; type: `object` or `nil`. The Simics Central client object used for distributing the simulation. (or a cell)

devices

Optional attribute; **read/write** access; type: `[[i|n,i,s,o|n,i]*]`. The connected devices. It is a list of entries, where each entry is a list [*global id*,*local id*,*name*,*obj*,*info*]. The *obj* is either a reference to the device object for local devices, or nil if the device was in another Simics process. list of device addresses

frequency

Optional attribute; **read/write** access; type: `integer`. The frequency of the clock used by the link to time stamp traffic and state changes. This defines the granularity of the time of any event on the link. The frequency is given as number of ticks per second.

last_frame

Pseudo attribute; **read/write** access; type: `[ibiiid]` or `nil`. This attribute contains the next frame to deliver, when read from a Generic_Message_Frame hap handler, [seq, handled, delivery_time, sender, address, frame]. It can also be set from the hap handler, which will modify the packet being sent. If it is set to nil or to an empty buffer, the packet will be dropped. From anywhere else than a Generic_Message_Frame hap handler, this attribute is nil and can't be set.

latency

Optional attribute; **read/write** access; type: `float` or `integer`. The latency on the link. Every packet that is sent over this link is delayed by at least the time specified in this attribute. Other simulated state changes, such as auto-negotiation and addresses also propagate over the link with the same delay. The type of this attribute is either an integer number of clock ticks in the time base specified in the *frequency* attribute, or a floating point number which defines the latency in seconds.

link_id

Pseudo attribute; **read/write** access; type: `integer`. The ID number of the link this link object belongs to.

link_obj_id

Pseudo attribute; **read/write** access; type: `integer`. The local ID number of this link object on the link.

linkname

Optional attribute; **read/write** access; type: `string` or `nil`. The global name of the link that this link objects belongs to. This name is used to identify links that are shared by several simulations. If this attribute is `nil`, the link can only be used locally.

master_obj_id

Pseudo attribute; **read/write** access; type: `integer`. The local ID number of the master link object on the link.

min_latency

Pseudo attribute; **read/write** access; type: `integer`. The minimum latency allowed. This is an integer number of clock ticks in the time base specified in the *frequency* attribute. In a distributed the value is given by Simics Central server. This attribute can not be changed.

nbytes

Optional attribute; **read/write** access; type: `integer`. The number of bytes transmitted over the network. This does not include any frames still in transfer.

next_seq

Optional attribute; **read/write** access; type: `integer`. The sequence number assigned to the next frame sent over the link

nframes

Optional attribute; **read/write** access; type: `integer`. The number of frames transmitted over the network. This does not include any frames still in transfer.

pending_frames

Optional attribute; **read/write** access; type: `[[ibi|[o[ii]|n]iid]*]`. Frames in transfer; [seq, handled, delivery_time, sender, address, frame]

generic-message-sample-device

Provided By
[generic-message-sample-device](#)

Interfaces Implemented
conf_object, [generic_message_device](#), [log_object](#)

Description
obsolete generic-message-link sample device

Attributes

address
Required attribute; **read/write** access; type: integer. Address of the device itself on the link.

link
Required attribute; **read/write** access; type: object. Link to connect to.

generic mmc card

Provided By
[generic mmc card](#)

Interfaces Implemented
conf_object, [log_object](#), mmc

Description
This is an IVP EMMC model in Simics

Command List

Commands
[info](#) print information about the device
[status](#) print status of the device

Command Descriptions

<generic mmc card>.info

Synopsis
[<generic mmc card>.info](#)

Description
Print detailed information about the configuration of the device.

<generic mmc card>.status

Synopsis
[<generic mmc card>.status](#)

Description
Print detailed information about the current status of the device.

generic_eth_phy

Provided By

[generic_eth_phy](#)

Interfaces Implemented

[conf_object](#), [ethernet_common](#), [ieee_802_3_phy](#), [ieee_802_3_phy_v2](#), [ieee_802_3_phy_v3](#),
[log_object](#), [mdio45_phy](#), [mii](#), [mii_management](#)

Ports

[mii_regs](#) ([int_register](#)), [mmd_auto_neg](#) ([int_register](#)), [mmd_default](#) ([int_register](#)), [mmd_dte_xs](#) ([int_register](#)), [mmd_mii_ext](#) ([int_register](#)), [mmd_pcs](#) ([int_register](#)), [mmd_phy_xs](#) ([int_register](#)), [mmd_pma_pmd](#) ([int_register](#)), [mmd_tc](#) ([int_register](#)), [mmd_vendor1](#) ([int_register](#)), [mmd_vendor2](#) ([int_register](#)), [mmd_wis](#) ([int_register](#))

Description

Represents a generic ethernet 802.3 PHY with MII and MDIO45 interfaces. It reflects link-status in the MII registers but does not autonegotiate speed with the link. All ethernet frames are passed through unmodified, except when loopback mode is enabled in the MII registers, then they are immediately returned back to the MAC. The **generic_eth_phy** can optionally model a maximum transmit bandwidth, see the *tx_bandwidth* attribute.

Attributes

address

Optional attribute; **read/write** access; type: `integer`. PHY identifier sent in calls to `ieee_802_3_mac` interface methods.

link

Optional attribute; **read/write** access; type: `[os], object, or nil`. The Ethernet link that the PHY is connected to or Nil if the PHY is disconnected. Must implement the `ethernet_common` interface.

link_led

Optional attribute; **read/write** access; type: `[os], object, or nil`. An object implementing the signal interface. It will ge a high signal when the link status is up.

loopback_delay

Optional attribute; **read/write** access; type: `float`. The time in seconds for sending back a frame when in loopback mode

mac

Optional attribute; **read/write** access; type: `[os], object, or nil`. Media access controller (MAC) object, implemenitng the `ieee_802_3_mac_v3` interface.

mii_regs_ext_status

Optional attribute; **read/write** access; type: `integer`. Default value of MII Extended Status (MII register address 15)

mii_regs_status

Optional attribute; **read/write** access; type: `integer`. Default value of MII Status (MII register address 1)

phy_id

Pseudo attribute; **read/write** access; type: `integer`. 32-bit PHY identifier

register_defaults

Optional attribute; **read/write** access; type: `[n|i{32}]`. Reset values of the MII registers. A `nil` value denotes that the standard reset value for that register is used.

registers

Pseudo attribute; **read/write** access; type: `[i{32}]`. The 32 MII management registers

tx_bandwidth

Optional attribute; **read/write** access; type: `integer`. The transmit bandwidth of the network interface in bits per second. The network interface will limit the rate at which it sends packets to remain below this bandwidth. Set to 0 for unlimited bandwidth.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

`<generic_eth_phy>.info`

Synopsis

`<generic_eth_phy>.info`

Description

Print detailed information about the configuration of the device.

`<generic_eth_phy>.status`

Synopsis

`<generic_eth_phy>.status`

Description

Print detailed information about the current status of the device.

generic_pcie_switch_port

Provided By

[generic-pcie-switch-port](#)

Interfaces Implemented

[bridge](#), [conf_object](#), [io_memory](#), [log_object](#), [pci_express](#), [pci_express_hotplug](#)

Ports

[pci_config](#) ([int_register](#), [io_memory](#))

Description

The generic_pcie_switch_port class implements a generic PCIe switch port.

The port is hotpluggable, handles I/O, memory and prefetchable transactions and has configurable vendor and device IDs. It does not implement any memory or I/O BARs. It exposes PCI Express 2.0 and MSI-64 capabilities.

Attributes

config_registers

Pseudo attribute; **read-only** access; type: `[i*]`. The PCI configuration registers, each 32 bits in size.

expansion_rom_size

Optional attribute; **read/write** access; type: `integer`. The size of the expansion ROM mapping.

is_upstream

Optional attribute; **read/write** access; type: `boolean`. Set to true for upstream port object

pci_bus

Optional attribute; **read/write** access; type: `[os]`, `object`, or `nil`. The PCI bus this device is connected to, implementing the `pci-bus` interface.

pci_config_command

Optional attribute; **read/write** access; type: `integer`. The PCI command register.

pci_config_device_id

Optional attribute; **read/write** access; type: `integer`. The Device ID of the PCI device

pci_config_vendor_id

Optional attribute; **read/write** access; type: `integer`. The Vendor ID of the PCI device

port_num

Optional attribute; **read/write** access; type: `integer`. Port number

secondary_bus

Optional attribute; **read/write** access; type: `[os]`, `object`, or `nil`. Secondary bus

Command List

Commands

info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
status	print status of the device

Command Descriptions

`<generic_pcie_switch_port>.info`

Synopsis

`<generic_pcie_switch_port>.info`

Description

Print detailed information about the configuration of the device.

`<generic_pcie_switch_port>.pci-header — deprecated`

Synopsis

`<generic_pcie_switch_port>.pci-header [-v]`

Description

This command is deprecated; use `<generic_pcie_switch_port>.print-pci-config-reg`s instead.

`<generic_pcie_switch_port>.print-pci-config-reg`

Synopsis

`<generic_pcie_switch_port>.print-pci-config-reg [-v]`

Description

Print the PCI configuration registers. The `-v` flag turns on verbose mode.

`<generic_pcie_switch_port>.status`

Synopsis

`<generic_pcie_switch_port>.status`

Description

Print detailed information about the current status of the device.

generic_spi_flash

Provided By

[generic-spi-flash](#)

Aliases

M25Pxx

Interfaces Implemented

[conf_object](#), [log_object](#), [serial_peripheral_interface_slave](#)

Ports

HRESET ([signal](#)), TOP_SECTOR_LOCK ([signal](#)), WRITE_PROTECT ([signal](#)), fcl ([int_register](#), [io_memory](#))

Description

The `generic_spi_flash` class implements the M25P05/10/20/40/80/16/32/64/128 serial flash memories, accessed via the SPI Interface. Different flash models using serial interface from different manufacturers can be implemented by manually configure JEDEC ID, sector number and sector size attributes.

Attributes

JEDEC_signature

Optional attribute; **read/write** access; type: `[iii]` or `nil`. JEDEC standard signature. It should be configured in format like [0x20, 0x20, 0x15] for M25P16. Please refer to the hardware documentation for the correct value.

elec_signature

Optional attribute; **read/write** access; type: `integer`. Electronic Signature, has default value 0x13. This is not the same as, or even a subset of, the JEDEC 16-bit Electronic Signature. This is for reason of backward compatibility, only, and should not be used for new designs. It should be configured with value 0x05 for M25P05-A, 0x10 for M25P10-A, 0x11 for M25P20, 0x12 for M25P40, 0x13 for M25P80, 0x14 for M25P16, 0x15 for M25P32. 0x16 for M25P64. For all other devices not listed above, please check the respective hardware documentation.

extended_id

Optional attribute; **read/write** access; type: `[ii]` or `nil`. Extended device identification. It is only implemented in some flash models. For those models, the two bytes are used to identify different devices. It should be configured like: [0x4d, 0x00] for S25FL032P. Please refer to the hardware documentation for the correct value.

mem_block

Required attribute; **read/write** access; type: `[os]` or `object`. Connects to an `image` object which holds the data, the `size` of attached object should be equal to `sector_number * sector_size`.

pp_buffer

Optional attribute; **read/write** access; type: `data` or `nil`. buffer for holding page program data

sector_lock

Optional attribute; **read/write** access; type: `data`. The SPM1 lock registers.

sector_number

Optional attribute; **read/write** access; type: `integer`. Sector number in the whole flash, has default value 16. Please refer to the hardware documentation for the correct value.

sector_size

Optional attribute; **read/write** access; type: `integer`. Bytes size for flash sector, has default value `0x10000(64 * 1024)`. Please refer to the hardware documentation for the correct value. For example, `0x8000` for M25P05-A

spi_master

Optional attribute; **read/write** access; type: `[os]`, `object`, or `nil`. SPI master this device connected to.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

`<generic_spi_flash>.info`

Synopsis

`<generic_spi_flash>.info`

Description

Print detailed information about the configuration of the device.

`<generic_spi_flash>.status`

Synopsis

`<generic_spi_flash>.status`

Description

Print detailed information about the current status of the device.

gfx-console

Provided By

[x11-console](#)

Aliases

x11-console

Interfaces Implemented

[abs_pointer_activate](#), [checkpoint](#), [conf_object](#), [log_object](#)

Description

The **graphics-console** class implements a graphical console with up to 256 colors, representing a computer screen. Several consoles may be present at any time. It also supports input of keyboard and mouse events. The objects of the **graphics-console** class should be connected to a graphics card device, and a keyboard/mouse device. The console shows its content in gray-scale when the simulation is stopped. The method to provide mouse input is specified with the *grab_button* and *grab_modifier* attributes.

Attributes

abs_pointer

Optional attribute; **read/write** access; type: `object` or `nil`. Absolute positioning pointer device connected to the console. Must implement the `abs_pointer` interface

abs_pointer_enabled

Optional attribute; **read/write** access; type: `boolean`. True if the absolute positioning pointer is enabled. The pointer can either be enabled statically in the script or from the device itself.

auto_release

Optional attribute; **read/write** access; type: `integer`. Flag whether the graphics-console automatically shall send a release event when the window focus is lost and keys still are pressed

break

Pseudo attribute; **read-only** access; **string** indexed; indexed type: `integer`. (*source_obj*, *filename*). Activate a graphical breakpoint previously saved to *filename*. Returns a breakpoint id.

clipboard_peer

Optional attribute; **read/write** access; type: `object` or `nil`. Clipboard peer object.

delete

Pseudo attribute; **write-only** access; type: `integer`. Delete breakpoint with given id.

depth

Optional attribute; **read/write** access; type: `integer`. The simulated depth.

double_x_limit

Optional attribute; **read/write** access; type: `integer`. If the width of the console is less than this value the width of every pixel will be doubled.

double_y_limit

Optional attribute; **read/write** access; type: `integer`. If the height of the console is less than this value the height of every pixel will be doubled.

fullscreen

Session attribute; **read/write** access; type: `integer`. Set fullscreen mode. The console will only run in fullscreen while input grabbing is active. Note that not all graphics-console implementations support fullscreen.

grab_button

Optional attribute; **read/write** access; type: `string`. One of left, middle and right. The grab button specifies which mouse button that is used to grab and ungrab input for the console.

grab_modifier

Optional attribute; **read/write** access; type: `string`. One of control, shift, alt and none. The modifier key used for grabbing and ungrabbing input for the console. When a modifier is selected, pressing only the middle mouse button cause the console to send this button event to the simulator. Currently only the left side modifier are interpreted.

input_enable

Optional attribute; **read/write** access; type: `integer`. Enable keyboard and mouse input to the console.

input_key_event

Pseudo attribute; **write-only** access; type: `[ii]`. Set to enter a key event. The first list element is 0 for key down, 1 for key up. The second list element is the key code.

kbd_type

Session attribute; **read/write** access; type: `string`. One of PC, PC_XI2, Sun, Mac or X11. By default the x11-console tries to find out what kind of keyboard that is connected to Simics, and use the raw key-codes from it. Since raw key-codes are used, the simulated machine must be setup with the same keyboard layout as the host. If the identification fails, Simics will use X11 cooked key symbols. The 'kbd_type' attribute allows manual setting of the keyboard type.

keyboard

Optional attribute; **read/write** access; type: `object` or `nil`. The console's keyboard device. Must implement the keyboard interface.

microm_x

Session attribute; **read/write** access; type: `integer`. Micrometers per horizontal pixel.

microm_y

Session attribute; **read/write** access; type: `integer`. Micrometers per vertical pixel.

minimize_window

Pseudo attribute; **write-only** access; type: `boolean`. By writing TRUE to this attribute, the graphics console window is minimized. A write of FALSE restores it again.

mouse

Optional attribute; **read/write** access; type: `object` or `nil`. The console's mouse device. Must implement the mouse interface

raise_window

Pseudo attribute; **write-only** access; type: `boolean`. By writing any value to this attribute, the graphics console window moved in front of other windows on the display.

recorder

Required attribute; **read/write** access; type: `object`. Recorder device for recording and playback of serial input.

refresh

Pseudo attribute; **write-only** access; type: `any`. If set the console will be completely redrawn.

rgb_colors

Optional attribute; **read/write** access; type: `[[iii]{256}]. ((r, g, b){256})` The color palette. 256 colors.

save_bmp

Pseudo attribute; **write-only** access; type: `string`. Save current console bitmap as a BMP file.

save_break

Pseudo attribute; **write-only** access; type: `[siiisi]. (source_obj, filename, left, top, right, bottom, comment)`. Saves a graphical breakpoint rectangle (top,left)->(bottom,right) to filename.

save_png

Pseudo attribute; **write-only** access; type: `[sii]`. Save current console bitmap as a PNG file. For additional integers are the width and height of the resulting image. They cannot be larger than the original image. If specified as 0, then the original image size is used.

title

Optional attribute; **read/write** access; type: `string`. The console window title.

try_open

Optional attribute; **read/write** access; type: `integer`. If set to 1, the window will be opened. Close with 0.

video

Optional attribute; **read/write** access; type: `object` or `nil`. The console's video device. Used to import video breakpoint functions. Must implement the `video_interface` interface.

visual_feedback

Optional attribute; **read/write** access; type: `boolean`. Enable or disable visual feedback.

x_size

Required attribute; **read/write** access; type: `integer`. The width of the console.

y_size

Required attribute; **read/write** access; type: `integer`. The height of the console.

Command List

Commands

<code>auto-release</code>	get/set auto-release flag
<code>break</code>	break on a graphics event specified by filename
<code>close</code>	close console window
<code>delete</code>	delete breakpoint
<code>disable-input</code>	ignore console input
<code>disable-visual-feedback</code>	disable visual feedback
<code>enable-input</code>	enable console input
<code>enable-visual-feedback</code>	enable visual feedback
<code>grab-setup</code>	set grab button and modifier
<code>input</code>	send string to a console
<code>open</code>	open console window
<code>refresh</code>	refresh console
<code>save-bmp</code>	save BMP image
<code>save-break-xy</code>	specify and save a graphical breakpoint
<code>save-png</code>	save PNG image
<code>switch-to-text-console</code>	replace graphics console with text console

Command Descriptions

`<gfx-console>.auto-release`

Synopsis

`<gfx-console>.auto-release [state]`

Description

Get or set the auto-release attribute on the gfx-console. If set, automatic release events will be sent for any keys still pressed when the window focus is lost.

```
<gfx-console>.break
```

Synopsis

```
<gfx-console>.break filename
```

Description

Activates a previously saved breakpoint specified by *filename* and returns a breakpoint id. When a graphical breakpoint is reached it is immediately deleted and a hap with id “**Gfx_Break_String**” is triggered. If no callbacks for this hap were registered Simics halts execution and returns to the command prompt.

```
<gfx-console>.close
```

Synopsis

```
<gfx-console>.close
```

Description

Close the console window.

See Also

[`<gfx-console>.open`](#)

```
<gfx-console>.delete
```

Synopsis

```
<gfx-console>.delete id
```

Description

Delete breakpoint given by *id*

```
<gfx-console>.disable-input
```

Synopsis

```
<gfx-console>.disable-input
```

Description

Ignore all inputs received from the console. Use this command to avoid sending input to the simulated machine if determinism is desired.

See Also

[`<gfx-console>.enable-input`](#)

```
<gfx-console>.disable-visual-feedback
```

Synopsis

```
<gfx-console>.disable-visual-feedback
```

Description

Disable visual feedback functionality in the console.

See Also

[`<gfx-console>.enable-visual-feedback`](#)

[`<gfx-console>.enable-input`](#)

Synopsis

[`<gfx-console>.enable-input`](#)

Description

Enable inputs from the console. New consoles will have input enabled.

See Also

[`<gfx-console>.disable-input`](#)

[`<gfx-console>.enable-visual-feedback`](#)

Synopsis

[`<gfx-console>.enable-visual-feedback`](#)

Description

Enable visual feedback functionality in the console.

See Also

[`<gfx-console>.disable-visual-feedback`](#)

[`<gfx-console>.grab-setup`](#)

Synopsis

[`<gfx-console>.grab-setup`](#) [“button”] [“modifier”]

Description

The grab button specifies which mouse button that is used to grab and ungrab input for the console. A keyboard modifier can also be specified that must be pressed for grabbing to occur. Valid buttons are left, middle and right, while valid modifiers are alt, control, shift and none. Only the left side modifier are currently used. When called with no arguments, the currently selected button and modifier will be printed.

[`<gfx-console>.input`](#)

Synopsis

[`<gfx-console>.input`](#) “*string*” [-e]

Description

Input string on console.

The -e switch enables emacs-style keystrokes. The following characters are accepted: 'C' (Control), 'A' (Alt), 'Del', 'Up', 'Down', 'Left', 'Right', 'Esc', 'F1' to 'F12' and 'Enter'. They can be combined using the '-' character. For example "C-a" will input Control and a pressed together. "C-A-Del" will produce the famous Control-Alt-Del sequence.

<gfx-console>.open

Synopsis

<gfx-console>.open

Description

Open the console window.

See Also

[`<gfx-console>.close`](#)

<gfx-console>.refresh

Synopsis

<gfx-console>.refresh

Description

Redraws all changes to the graphics console.

<gfx-console>.save-bmp

Synopsis

<gfx-console>.save-bmp *filename*

Description

This command saves the current console window contents to an image file in BMP format.

<gfx-console>.save-break-xy

Synopsis

<gfx-console>.save-break-xy *filename left top right bottom* [“*comment*”]

Description

Lets the user select a rectangular area inside the graphics console by specifying the top left corner (*left, top*) and the bottom right corner (*right, bottom*). The selected area will be saved as a binary graphical breakpoint file. An optional comment may be added that will be put in the beginning of the file.

```
<gfx-console>.save-png
```

Synopsis

```
<gfx-console>.save-png filename [width] [height]
```

Description

This command saves the current console window contents to an image file in PNG format. It can be scaled down if width and height parameters are given.

```
<gfx-console>.switch-to-text-console
```

Synopsis

```
<gfx-console>.switch-to-text-console
```

Description

Replace the graphics with a text console.

gui

Provided By

[Simics Core](#)

Interfaces Implemented

[conf_object](#), [log_object](#)

Description

Class used by the built-in GUI to communicate with remote frontend handler in Simics.

hap-meter

Provided By
[hap-meter](#)

Interfaces Implemented
conf_object, [log_object](#)

Description
Hap meter

Attributes

blip

Pseudo attribute; **write-only** access; type: string. Write a stream name (any string) to this attribute to register a blip on that stream.

Command List

Commands

info	print information about the device
listen-for-exceptions	listen for exception haps
listen-for-hap	listen for a specified hap
status	print status of the device

Command Descriptions

<hap-meter>.info

Synopsis

<hap-meter>.info

Description

Print detailed information about the configuration of the device.

<hap-meter>.listen-for-exceptions

Synopsis

<hap-meter>.listen-for-exceptions [processor] ["exception"]

Description

Instruct the hap meter to listen for exceptions on the given processor (or all processors, if none is given). You may restrict the listening to a specific type of exception; if you do not, all exceptions will be collected.

See Also

[new-hap-meter](#)

`<hap-meter>.listen-for-hap`

Synopsis

`<hap-meter>.listen-for-hap "hap" [object]`

Description

Instruct the hap meter to listen for a certain hap. You may restrict the listening to a specific object; if you do not, haps triggered by all objects will be collected.

See Also

[new-hap-meter](#)

`<hap-meter>.status`

Synopsis

`<hap-meter>.status`

Description

Print detailed information about the current status of the device.

host-condor-pci-1553

Provided By

[host-condor-pci-1553](#)

Interfaces Implemented

conf_object, [log_object](#), ms1553_bridge_bus

Description

The **host-condor-pci-1553** class is a layer between RTs in Simics and a host Condor PCI 1553 card. The RTs in Simics are emulated by the host card. It is not possible for the RTs in Simics to control message transfers directly in real time. This is due to hard timing restrictions for the real MIL-STD-1553 bus connected to the host card. The emulated RTs (shadow RTs) are set up by the RTs in Simics to make them respond to messages as they would have done. The RTs in Simics get a copy of the messages handled by the shadow RTs. The **host-condor-pci-1553** class sets up the host card when new instance of the class is created. It creates a emulated RT when a Simics RT connect to it. The class includes the possibility to test RTs in Simics by emulating a simple bus controller. The bus controller is setup using the *bc_test* attribute. This class requires the Condor BusTools 1553-API to be installed on the host to work. Only one instance of this component is allowed in a simulation session.

host-pmc-1553

Provided By
[host-pmc-1553](#)

Interfaces Implemented
[conf_object](#), [log_object](#), [ms1553_link](#)

Description

The host-pmc-1553 provides a connection from a simulated 1553 Bus Controller to a real Alphi PMC-1553 device, allowing the simulated controller to access devices on a real MIL-STD-1553 bus. The host-pmc-1553 object should be connected directly to the simulated 1553 bus controller device in place of the ms1553-link.

To compile the host-pmc-1553 module, a driver development kit from Alphi Technology is required. The Alphi Technology PMC 1553 driver, needed to use this module, supports Windows 2000 and has also been tested on Windows Server 2003 SE.

host-serial-console

Provided By
[host-serial-console](#)

Aliases
pty-console

Interfaces Implemented
[conf_object](#), [log_object](#), [serial_device](#)

Description
The **host-serial-console** class provides a pseudo terminal access to the target's pty.

Attributes

existing_port

Optional attribute; **read/write** access; type: `string` or `nil`. Name of an existing host pty to use instead of allocating a new pseudo one.

recorder

Required attribute; **read/write** access; type: `object`. Recorder device for recording and playback of serial input.

Command List

Commands

switch-to-serial-link	<i>deprecated</i> —
switch-to-telnet-console	<i>deprecated</i> —
switch-to-text-console	<i>deprecated</i> —

Command Descriptions

`<host-serial-console>.switch-to-serial-link — deprecated`

Synopsis

`<host-serial-console>.switch-to-serial-link "linkname"`

Description

This command is deprecated; use [a manual component connection](#) to a serial link instead.

`<host-serial-console>.switch-to-telnet-console — deprecated`

Synopsis

`<host-serial-console>.switch-to-telnet-console [port]`

Description

This command is deprecated; use `<std-host-serial-console>.switch-to-telnet-console` or `<std-text-console>.switch-to-telnet-console` instead.

<host-serial-console>.switch-to-text-console — *deprecated*

Synopsis

`<host-serial-console>.switch-to-text-console`

Description

This command is deprecated; use `<std-telnet-console>.switch-to-text-console` or `<std-host-serial-console>.switch-to-text-console` instead.

hostfs

Provided By

[hostfs](#)

Interfaces Implemented

[conf_object](#), [io_memory](#), [log_object](#)

Description

The hostfs device is used in conjunction with a target OS module to allow the simulated machine to access the filesystem of the host machine. In order to work, the hostfs-device needs to be mapped into a specific location in the physical memory space. This address is target OS and architecture dependent.

Command List

Commands

- info** print information about the device
- root** get or set the hostfs root directory
- status** print status of the device

Command Descriptions

<hostfs>.info

Synopsis

<hostfs>.info

Description

Print detailed information about the configuration of the device.

<hostfs>.root

Synopsis

<hostfs>.root [dir]

Description

If *dir* is specified, set the host directory that is visible to the simulated machine accordingly. This is only allowed when the file system is not mounted by the simulated machine.

Returns the current root directory.

<hostfs>.status

Synopsis

<hostfs>.status

Description

Print detailed information about the current status of the device.

hypersim-pattern-matcher

Provided By

[hypersim-pattern-matcher](#)

Interfaces Implemented

conf_object, hypersim_pattern_matcher, [log_object](#)

Description

Framework for instruction pattern detection and protection. This class requires other classes, which define the patterns to be detected and uses this class, to be any useful. Instruction patterns can be defined to identify specific idle loops, busy-wait loops, spinlocks etc in memory which can be fast forwarded in the simulators time

Attributes

memory_space

Required attribute; **read/write** access; type: `object`. Physical memory space.

Command List

Commands

- | | |
|--------------------------------|------------------------------------|
| delete-pattern | Remove a pattern from the matcher |
| info | print information about the device |
| status | print status of the device |

Command Descriptions

<hypersim-pattern-matcher>.delete-pattern

Synopsis

`<hypersim-pattern-matcher>.delete-pattern arg1`

Description

Remove a pattern from the pattern-matcher. The *pattern* object name is used to identify the pattern to remove.

<hypersim-pattern-matcher>.info

Synopsis

`<hypersim-pattern-matcher>.info`

Description

Print detailed information about the configuration of the device.

<hypersim-pattern-matcher>.status

Synopsis

<hypersim-pattern-matcher>.status

Description

Print detailed information about the current status of the device.

i21150

Provided By

[i21150](#)

Interfaces Implemented

[bridge](#), [conf_object](#), [io_memory](#), [log_object](#)

Description

The i21150 object models the Intel® i21150 PCI to PCI Bridge

Attributes

secondary_bus

Required attribute; **read/write** access; type: `object`. Secondary (downstream) PCI bus.

Command List

Commands

info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
status	print status of the device

Command Descriptions

`<i21150>.info`

Synopsis

`<i21150>.info`

Description

Print detailed information about the configuration of the device.

`<i21150>.pci-header — deprecated`

Synopsis

`<i21150>.pci-header [-v]`

Description

This command is deprecated; use [`<i21150>.print-pci-config-reg`](#) instead.

`<i21150>.print-pci-config-reg`

Synopsis

`<i21150>.print-pci-config-reg [-v]`

Description

Print the PCI configuration registers. The *-v* flag turns on verbose mode.

```
<i21150>.status
```

Synopsis

```
<i21150>.status
```

Description

Print detailed information about the current status of the device.

i21152

Provided By

[i21152](#)

Interfaces Implemented

[bridge](#), [conf_object](#), [io_memory](#), [log_object](#)

Description

The i21152 object models the Intel® i21152 PCI to PCI Bridge

Attributes

secondary_bus

Required attribute; **read/write** access; type: `object`. Secondary (downstream) PCI bus.

Command List

Commands

info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
status	print status of the device

Command Descriptions

`<i21152>.info`

Synopsis

`<i21152>.info`

Description

Print detailed information about the configuration of the device.

`<i21152>.pci-header — deprecated`

Synopsis

`<i21152>.pci-header [-v]`

Description

This command is deprecated; use [`<i21152>.print-pci-config-reg`](#) instead.

`<i21152>.print-pci-config-reg`

Synopsis

`<i21152>.print-pci-config-reg [-v]`

Description

Print the PCI configuration registers. The *-v* flag turns on verbose mode.

```
<i21152>.status
```

Synopsis

```
<i21152>.status
```

Description

Print detailed information about the current status of the device.

i21154

Provided By

[i21154](#)

Interfaces Implemented

[bridge](#), [conf_object](#), [io_memory](#), [log_object](#)

Ports

`pci_config (int_register, io_memory)`

Description

Intel 21154 PCI-PCI bridge. Not implemented:

- VGA mode
- ISA mode

Attributes

config_registers

Pseudo attribute; **read-only** access; type: `[i*]`. The PCI configuration registers, each 32 bits in size.

expansion_rom_size

Optional attribute; **read/write** access; type: `integer`. The size of the expansion ROM mapping.

pci_bus

Required attribute; **read/write** access; type: `[os] or object`. The PCI bus this device is connected to, implementing the `pci-bus` interface.

pci_config_command

Optional attribute; **read/write** access; type: `integer`. The PCI command register.

pci_config_device_id

Optional attribute; **read/write** access; type: `integer`. The Device ID of the PCI device

pci_config_vendor_id

Optional attribute; **read/write** access; type: `integer`. The Vendor ID of the PCI device

secondary_bus

Required attribute; **read/write** access; type: `[os] or object`. Secondary bus

Command List

Commands

info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
status	print status of the device

Command Descriptions

`<i21154>.info`

Synopsis

`<i21154>.info`

Description

Print detailed information about the configuration of the device.

`<i21154>.pci-header — deprecated`

Synopsis

`<i21154>.pci-header [-v]`

Description

This command is deprecated; use `<i21154>.print-pci-config-reg`s instead.

`<i21154>.print-pci-config-reg`s

Synopsis

`<i21154>.print-pci-config-reg [-v]`

Description

Print the PCI configuration registers. The `-v` flag turns on verbose mode.

`<i21154>.status`

Synopsis

`<i21154>.status`

Description

Print detailed information about the current status of the device.

i21554-prim

Provided By

[i21554](#)

Interfaces Implemented

[bridge](#), [conf_object](#), [io_memory](#), [log_object](#)

Description

The Intel® i21554-prim models the primary interface on the PCI to PCI bridge

Attributes

scnd_interface

Required attribute; **read/write** access; type: `object`. Secondary interface object for the bridge

i21554-scnd

Provided By

[i21554](#)

Interfaces Implemented

[bridge](#), [conf_object](#), [io_memory](#), [log_object](#)

Description

The i21554-scnd models the secondary interface on the PCI to PCI bridge

Attributes

prim_interface

Required attribute; **read/write** access; type: `object`. Primary interface object for the bridge

i21555-prim

Provided By

[i21555](#)

Interfaces Implemented

[bridge](#), [conf_object](#), [io_memory](#), [log_object](#)

Description

The Intel® i21555-prim models the primary interface on the PCI to PCI bridge

Attributes

scnd_interface

Required attribute; **read/write** access; type: `object`. Secondary interface object for the bridge

i21555-scnd

Provided By

[i21555](#)

Interfaces Implemented

[bridge](#), [conf_object](#), [io_memory](#), [log_object](#)

Description

The i21555-scnd models the secondary interface on the PCI to PCI bridge

Attributes

prim_interface

Required attribute; **read/write** access; type: `object`. Primary interface object for the bridge

i2c-bus

Provided By
[i2c-bus](#)

Interfaces Implemented

`conf_object`, [i2c_bridge](#), [i2c_bus](#), [i2c_master](#), [i2c_slave](#), [log_object](#)

Ports

SCL ([signal](#)), SDA ([signal](#))

Description

The i2c-bus class implements the functionality of an I2C bus, to which several I2C devices can connect.

Attributes

current_slave

Optional attribute; **read/write** access; type: `object` or `nil`. The slave of the currently ongoing transaction, if any.

current_state

Optional attribute; **read/write** access; type: `integer`. The current state of the bus, where 0 is idle, 1 is master transmit, 2 is master receive, 3 is slave transmit, and 4 is slave receive.

first_read_request

Optional attribute; **read/write** access; type: `boolean`. When forwarding a connection to a slave device via an `i2c_link` object and in slave transmit mode, this attribute is true right after the start condition and before the first read request. Marks that the next read should not be preceded by an `ack_read_request`

i2c_devices

Pseudo attribute; **read-only** access; type: `[[si]*]`. Contains a list of the devices connected to the bus. Each list entry is a list of (`device`, `address`) for each address that has a device connected. Devices connect themselves using the `i2c_bus` interface.

i2c_link

Optional attribute; **read/write** access; type: `object` or `nil`. If set, the bus will try to share all communication with this i2c link object. This is useful when converting i2c devices to use the new interfaces `i2c_link`, `i2c_slave` and `i2c_master`, since it allows devices to be converted one by one. All connected slave devices are forwarded as slave devices in this i2c link object. Slave devices registered on the link will be accessible from master devices on this i2c bus, but only as long as all responses arrive immediately, i.e., as long as both the link and the slave device have zero latency.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<i2c-bus>.info

Synopsis

<i2c-bus>.info

Description

Print detailed information about the configuration of the device.

<i2c-bus>.status

Synopsis

<i2c-bus>.status

Description

Print detailed information about the current status of the device.

i2c-link-endpoint

Provided By
[i2c-link-v2](#)

Interfaces Implemented
conf_object, [i2c_master_v2](#), [i2c_slave_v2](#), log_object

Description
I2C link v2 endpoint

Attributes

device

Required attribute; **read/write** access; type: [os], object, or nil. The device connected to this endpoint.

id

Required attribute; **read/write** access; type: integer. Endpoint ID. The ID of each endpoint must be unique among the link's endpoints, and it may not be 0 or 0xfffffffffffffff

link

Required attribute; **read/write** access; type: object. The link object to which this endpoint belongs.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<i2c-link-endpoint>.info

Synopsis

<i2c-link-endpoint>.info

Description

Print detailed information about the configuration of the device.

<i2c-link-endpoint>.status

Synopsis

<i2c-link-endpoint>.status

Description

Print detailed information about the current status of the device.

i2c-link-impl

Provided By
[i2c-link-v2](#)

Interfaces Implemented
conf_object, [log_object](#)

Description
I2C link v2

Attributes

goal_latency
Required attribute; **read/write** access; type: float. The desired latency for this link.

Command List

Commands
info print information about the device
status print status of the device

Command Descriptions

`<i2c-link-impl>.info`

Synopsis
`<i2c-link-impl>.info`

Description
Print detailed information about the configuration of the device.

`<i2c-link-impl>.status`

Synopsis
`<i2c-link-impl>.status`

Description
Print detailed information about the current status of the device.

i2c_link_v1

Provided By

[i2c-link-v1](#)

Aliases

i2c_link

Interfaces Implemented

conf_object, [i2c_link](#), [i2c_master_v2](#), [i2c_slave_v2](#), [log_object](#)

Description

The i2c_link_v1 class implements the functionality of an I2C bus, to which several I2C devices can connect. Connecting I2C devices should implement the `i2c_master` or `i2c_slave` interface.

Attributes

bus_freed_queue

Optional attribute; **read/write** access; type: `[o*]`. List of i2c master devices waiting for a `bus_freed` call. Index 0 is first in the queue.

enqueued_start

Optional attribute; **read/write** access; type: `integer`. When connected as slave to an i2c-link-v2 object, a stop and a start request may arrive while waiting for an `ack_read_response` from a slave. This attribute stores the start address, -1 if no start is enqueued.

link_state

Optional attribute; **read/write** access; type: `string`. Internal state of the link

link_v2

Optional attribute; **read/write** access; type: `[os], object, or nil`. This connect is used to connect with Link v2. It should only setby the component

slave_devices

Pseudo attribute; **read-only** access; type: `[[oi]*]`. Contains a list of the slave devices connected to the link. Each list entry is a list of (`device, address`) for each address that has a device connected. Devices connect themselves using the `i2c_link` interface.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<i2c_link_v1>.info

Synopsis

<i2c_link_v1>.info

Description

Print detailed information about the configuration of the device.

<i2c_link_v1>.status

Synopsis

<i2c_link_v1>.status

Description

Print detailed information about the current status of the device.

i2c_slave_v2_to_bus_adapter

Provided By

[i2c-bus](#)

Interfaces Implemented

`conf_object, i2c_device, i2c_master_v2, log_object`

Description

The `i2c_slave_v2_to_bus_adapter` class makes it possible to connect an I2C slave device, implementing `i2c_slave_v2` interface, directly to a legacy `i2c-bus` object. The slave device must send all responses synchronously, i.e., before a `start()`, `write()` or `read()` call in the `i2c_slave_v2` interface has returned, the slave it must have called the corresponding `acknowledge()` or `read_response()` method. The adapter can be used on a single-master bus where the master device has not yet been upgraded to the new `i2c_master_v2/i2c_slave_v2` interfaces, but one of the slave devices has.

Attributes

`i2c_bus`

Required attribute; **read/write** access; type: `[os]` or `object`. I2C bus to connect to.

`i2c_slave`

Required attribute; **read/write** access; type: `[os]` or `object`. Simple I2C slave device connected with.

Command List

Commands

`info` print information about the device

`status` print status of the device

Command Descriptions

`<i2c_slave_v2_to_bus_adapter>.info`

Synopsis

`<i2c_slave_v2_to_bus_adapter>.info`

Description

Print detailed information about the configuration of the device.

`<i2c_slave_v2_to_bus_adapter>.status`

Synopsis

`<i2c_slave_v2_to_bus_adapter>.status`

Description

Print detailed information about the current status of the device.

i2c_wire

Provided By

[i2c-link-v2](#)

Interfaces Implemented

conf_object, [log_object](#)

Ports

i2c_port[2] ([i2c_master_v2](#), [i2c_slave_v2](#))

Description

The i2c_wire class implements bridge functionality between two i2c-link-v2 endpoints. It will forward everything from one side to another without analyzing. To use this device, connect i2c_link[0]/i2c_port[0] to one endpoint and i2c_link[1]/i2c_port[1] to another endpoint

Attributes

i2c_link_v2

Optional attribute; **read/write** access; type: [o|[os]|n{2}]. These connectors are used to connect two i2c-link-v2 objects.

Command List

Commands

info print information about the device

status print status of the device

Command Descriptions

<i2c_wire>.info

Synopsis

<i2c_wire>.info

Description

Print detailed information about the configuration of the device.

<i2c_wire>.status

Synopsis

<i2c_wire>.status

Description

Print detailed information about the current status of the device.

i82543

Provided By

[i82543](#)

Interfaces Implemented

`conf_object`, `ethernet_common`, `ethernet_device`, `io_memory`, `log_object`

Description

This class models the Intel® 82543 Gigabit Ethernet Controller. Limitations on all i8254x:

- flow control functionality is missing. FCAH, FCT, FCRTH, FCRTL, and FCTTV (bug 1107)
- test mode is missing. FTR0-7, FCR, and TRC. (bug 1108)
- ACPI/Power management is missing. (bug 1109)
- FIFO related diagnostic registers are missing. RDFH, RDFT, TDFH. (bug 1110)
- config data should be read from eeprom, and not hardcoded. (bug 1111)
- The transmit-interrupt-delay (TIDV) is unimplemented (bug 1125)
- Receive descriptors with NULL data pointers not supported (bug 1623)

Attributes

add_crc_on_inject

Optional attribute; **read/write** access; type: `integer`. Frames injected using the 'inject_packet' will get a correctly calculated CRC added at the end when this attribute is set to 1 (default). When set to 0, the user has to supply a CRC field with the injected frame. Note that you must always provide room for the CRC field, even when this attribute is set to 1.

eeprom

Required attribute; **read/write** access; type: `object`. Serial 4-wire EEPROM.

inject_packet

Pseudo attribute; **write-only** access; type: `data`. Attribute used to send a packet to the network device. Writing this attribute at any time injects a new packet into the device (without involving the network simulation). Injecting a packet copies the packet data, allowing the caller to reuse or dispose of the buffer used for creating the packet, after the attribute is written.

last_frame

Pseudo attribute; **read/write** access; type: `data` or `nil`. The frame that is currently about to be sent or received. The format is a raw Ethernet frame. This attribute is only valid in `Ethernet_Transmit` and `Ethernet_Receive` hap callbacks. It is possible to override the packet by assigning this attribute. The device will drop the current packet if the attribute is set to Nil, or with data of zero length.

link

Optional attribute; **read/write** access; type: `object` or `nil`. The Ethernet link that the network device is connected to.

model_crc

Optional attribute; **read/write** access; type: `integer`. If set to 1, the device will calculate and include an Ethernet CRC on all transmitted frames, and check the CRC on received frames. This attribute is **ignored** when new-style links are used.

tx_bandwidth

Optional attribute; **read/write** access; type: `integer`. The transmit bandwidth of the network interface in bits per second. The network interface will limit the rate at which it sends packets to remain below this bandwidth. Set to 0 for unlimited bandwidth.

Command List

Commands

connect	<i>deprecated</i> — connect to a simulated Ethernet link
disconnect	<i>deprecated</i> — disconnect from simulated link
info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
status	print status of the device

Command Descriptions

`<i82543>.connect — deprecated`

Synopsis

`<i82543>.connect [-auto] link`

Description

This command is deprecated; use [connect](#) instead.

Connect the device to a simulated Ethernet link. The flag '`-auto`' is deprecated and shouldn't be used.

See Also

[<i82543>.disconnect](#)

`<i82543>.disconnect — deprecated`

Synopsis

`<i82543>.disconnect`

Description

This command is deprecated; use [disconnect](#) instead.

Disconnect the device from a simulated Ethernet link.

See Also

[`<i82543>.connect`](#)

<i82543>.info**Synopsis**

[`<i82543>.info`](#)

Description

Print detailed information about the configuration of the device.

<i82543>.pci-header — *deprecated***Synopsis**

[`<i82543>.pci-header \[-v\]`](#)

Description

This command is deprecated; use [`<i82543>.print-pci-config-reg`](#)s instead.

<i82543>.print-pci-config-reg**Synopsis**

[`<i82543>.print-pci-config-reg \[-v\]`](#)

Description

Print the PCI configuration registers. The `-v` flag turns on verbose mode.

<i82543>.status**Synopsis**

[`<i82543>.status`](#)

Description

Print detailed information about the current status of the device.

i82546

Provided By

[i82546](#)

Interfaces Implemented

`conf_object`, `ethernet_common`, `ethernet_device`, `io_memory`, `log_object`

Description

This class models the Intel® 82546 Gigabit Ethernet Controller. Limitations on all i8254x:

- flow control functionality is missing. FCAH, FCT, FCRTH, FCRTL, and FCTTV (bug 1107)
- test mode is missing. FTR0-7, FCR, and TRC. (bug 1108)
- ACPI/Power management is missing. (bug 1109)
- FIFO related diagnostic registers are missing. RDFH, RDFT, TDFH. (bug 1110)
- config data should be read from eeprom, and not hardcoded. (bug 1111)
- The transmit-interrupt-delay (TIDV) is unimplemented (bug 1125)
- Receive descriptors with NULL data pointers not supported (bug 1623)

Attributes

add_crc_on_inject

Optional attribute; **read/write** access; type: `integer`. Frames injected using the 'inject_packet' will get a correctly calculated CRC added at the end when this attribute is set to 1 (default). When set to 0, the user has to supply a CRC field with the injected frame. Note that you must always provide room for the CRC field, even when this attribute is set to 1.

eeprom

Required attribute; **read/write** access; type: `object`. Serial 4-wire EEPROM.

inject_packet

Pseudo attribute; **write-only** access; type: `data`. Attribute used to send a packet to the network device. Writing this attribute at any time injects a new packet into the device (without involving the network simulation). Injecting a packet copies the packet data, allowing the caller to reuse or dispose of the buffer used for creating the packet, after the attribute is written.

last_frame

Pseudo attribute; **read/write** access; type: `data` or `nil`. The frame that is currently about to be sent or received. The format is a raw Ethernet frame. This attribute is only valid in `Ethernet_Transmit` and `Ethernet_Receive` hap callbacks. It is possible to override the packet by assigning this attribute. The device will drop the current packet if the attribute is set to Nil, or with data of zero length.

link

Optional attribute; **read/write** access; type: `object` or `nil`. The Ethernet link that the network device is connected to.

model_crc

Optional attribute; **read/write** access; type: `integer`. If set to 1, the device will calculate and include an Ethernet CRC on all transmitted frames, and check the CRC on received frames. This attribute is **ignored** when new-style links are used.

tx_bandwidth

Optional attribute; **read/write** access; type: `integer`. The transmit bandwidth of the network interface in bits per second. The network interface will limit the rate at which it sends packets to remain below this bandwidth. Set to 0 for unlimited bandwidth.

Command List

Commands

connect	<i>deprecated</i> — connect to a simulated Ethernet link
disconnect	<i>deprecated</i> — disconnect from simulated link
info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
status	print status of the device

Command Descriptions

`<i82546>.connect — deprecated`

Synopsis

`<i82546>.connect [-auto] link`

Description

This command is deprecated; use [connect](#) instead.

Connect the device to a simulated Ethernet link. The flag '`-auto`' is deprecated and shouldn't be used.

See Also

[<i82546>.disconnect](#)

`<i82546>.disconnect — deprecated`

Synopsis

`<i82546>.disconnect`

Description

This command is deprecated; use [disconnect](#) instead.

Disconnect the device from a simulated Ethernet link.

See Also

[`<i82546>.connect`](#)

<i82546>.info**Synopsis**

[`<i82546>.info`](#)

Description

Print detailed information about the configuration of the device.

<i82546>.pci-header — *deprecated***Synopsis**

[`<i82546>.pci-header \[-v\]`](#)

Description

This command is deprecated; use [`<i82546>.print-pci-config-reg`](#)s instead.

<i82546>.print-pci-config-reg**Synopsis**

[`<i82546>.print-pci-config-reg \[-v\]`](#)

Description

Print the PCI configuration registers. The `-v` flag turns on verbose mode.

<i82546>.status**Synopsis**

[`<i82546>.status`](#)

Description

Print detailed information about the current status of the device.

i82559

Provided By

[i82559](#)

Interfaces Implemented

conf_object, [ieee_802_3_mac](#), [ieee_802_3_mac_v3](#), [io_memory](#), [log_object](#)

Ports

csr ([int_register](#), [io_memory](#)), pci_config ([int_register](#), [io_memory](#))

Description

Intel® 82559 Fast Ethernet Controller.

Attributes

config_registers

Pseudo attribute; **read-only** access; type: [i*]. The PCI configuration registers, each 32 bits in size.

expansion_rom_size

Optional attribute; **read/write** access; type: integer. The size of the expansion ROM mapping.

mii

Required attribute; **read/write** access; type: [os] or object. Object connected to the MII management interface.

pci_bus

Required attribute; **read/write** access; type: [os] or object. The PCI bus this device is connected to, implementing the pci-bus interface.

pci_config_command

Optional attribute; **read/write** access; type: integer. The PCI command register.

pci_config_device_id

Optional attribute; **read/write** access; type: integer. The Device ID of the PCI device

pci_config_vendor_id

Optional attribute; **read/write** access; type: integer. The Vendor ID of the PCI device

phy

Required attribute; **read/write** access; type: [os] or object. Phy object

phy_address

Required attribute; **read/write** access; type: integer. The PHY address of which object mii connects to

serial_eeprom

Required attribute; **read/write** access; type: [os] or object. Serial EEPROM

Command List

Commands

info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
status	print status of the device

Command Descriptions

`<i82559>.info`

Synopsis

`<i82559>.info`

Description

Print detailed information about the configuration of the device.

`<i82559>.pci-header — deprecated`

Synopsis

`<i82559>.pci-header [-v]`

Description

This command is deprecated; use `<i82559>.print-pci-config-reg`s instead.

`<i82559>.print-pci-config-reg`s

Synopsis

`<i82559>.print-pci-config-reg [-v]`

Description

Print the PCI configuration registers. The `-v` flag turns on verbose mode.

`<i82559>.status`

Synopsis

`<i82559>.status`

Description

Print detailed information about the current status of the device.

id-splitter

Provided By

[id-splitter](#)

Interfaces Implemented

[conf_object](#), [log_object](#), [timing_model](#)

Description

The id splitter module splits up memory operations into separate data and instruction streams. Data operations are forwarded to the timing interface of the object specified by the dbranch attribute and, in the same manner, instruction operations are forwarded to the ibranch.

ide

Provided By

[ide](#)

Interfaces Implemented

`conf_object`, [io_memory](#), [log_object](#)

Ports

`DMA_DACK` ([signal](#)), `DMA_DDONE` ([signal](#))

Description

The IDE class implements the functionality of an IDE controller and its interaction with the DMA controller. Transfer timing (command start -> interrupt) for DMA is properly modelled.

Attributes

bus_master_dma

Optional attribute; **read/write** access; type: `object` or `nil`. DMA controller implementing bus-master-ide.

dma_dack_level

Optional attribute; **read/write** access; type: `boolean`. DMA controller implementing signal interface. Upon a DMA request, the DREQ signal will be held high.

dma_dreq_target

Optional attribute; **read/write** access; type: `[os]`, `object`, or `nil`. DMA controller implementing signal interface. Upon a DMA request, the DREQ signal will be held high.

dma_ready

Optional attribute; **read/write** access; type: `integer`. Non-zero if the DMA controller is ready for transfer.

interrupt_delay

Optional attribute; **read/write** access; type: `float`. Time in seconds between state changes.

interrupt_pin

Optional attribute; **read/write** access; type: `integer`. Interrupt request status.

irq_dev

Optional attribute; **read/write** access; type: `object` or `nil`. Interrupt target implementing the simple_interrupt interface.

irq_level

Required attribute; **read/write** access; type: `integer`. Interrupt level.

master

Optional attribute; **read/write** access; type: `object` or `nil`. Master device. Must implement the ide-device interface.

model_dma_delay

Optional attribute; **read/write** access; type: `integer`. If set, the controller will accurately model DMA transfer bandwidth (infinite bandwidth otherwise).

primary

Optional attribute; **read/write** access; type: `integer`. Set to one if this is a primary IDE controller.

slave

Optional attribute; **read/write** access; type: `object` or `nil`. Slave device. Must implement the ide-device interface.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<ide>.info

Synopsis

<ide>.info

Description

Print detailed information about the configuration of the device.

<ide>.status

Synopsis

<ide>.status

Description

Print detailed information about the current status of the device.

ide-cdrom

Provided By

[ide](#)

Interfaces Implemented

[conf_object](#), [log_object](#)

Description

The **ide-cdrom** class models a single slot removable IDE/ATAPI CD-ROM.

Attributes

cd_media

Optional attribute; **read/write** access; type: `object` or `nil`. Device media implementing `cdrom_media`. Eject media by setting this attribute to `NIL`.

firmware_id

Optional attribute; **read/write** access; type: `string`. Device firmware id. Up to 8 character long string.

geometry_valid

Optional attribute; **read/write** access; type: `integer`. Geometry valid.

locked

Optional attribute; **read/write** access; type: `integer`. Media locked.

media_change

Optional attribute; **read/write** access; type: `integer`. Media has changed since the last `GET_MEDIA_STATUS` command.

model_id

Optional attribute; **read/write** access; type: `string`. Device model id. Up to 40 character long string.

power_mode

Optional attribute; **read/write** access; type: `integer`. Device power mode (0==active, 1==idle, 2==standby, 3==sleep).

serial_number

Optional attribute; **read/write** access; type: `string`. Device serial number. Up to 20 character long string.

stat_atapi_commands

Optional attribute; **read/write** access; type: $[i \{256\}] \cdot (f_0, \dots, f_{255})$. Executed ATAPI command count (command i has been issued f_i times).

stat_bytes_read

Optional attribute; **read/write** access; type: **integer**. Total number of bytes read from device using PIO.

stat_bytes_transferred_dma

Optional attribute; **read/write** access; type: **integer**. Total number of bytes read or written to or from device using DMA.

stat_bytes_written

Optional attribute; **read/write** access; type: **integer**. Total number of bytes written to device using PIO.

stat_commands

Optional attribute; **read/write** access; type: $[i\{256\}] \cdot (f_0, \dots, f_{255})$. Executed command count (command *i* has been issued *f_i* times).

Command List

Commands

- eject** eject media from CD-ROM drive
- info** print information about the device
- insert** insert media in CD-ROM drive

Command Descriptions

<ide-cdrom>.eject

Synopsis

<ide-cdrom>.eject

Description

Eject a media from the CD-ROM drive. The media must have been previously inserted with the 'insert' command.

<ide-cdrom>.info

Synopsis

<ide-cdrom>.info

Description

Print detailed information about the configuration of the device.

<ide-cdrom>.insert

Synopsis

<ide-cdrom>.insert *media* [-cd] [-dvd]

Description

Insert a media in the CD-ROM drive. The media is the name of a CD-ROM media object, e.g. a file-*cdrom*. Use the *-cd* flag to indicate that the media is a CD-ROM, or the *-dvd* flag for DVD-ROM. CD-ROM is the default choice if none of the flags are given.

ide-disk

Provided By

[ide](#)

Interfaces Implemented

[conf_object](#), [log_object](#)

Description

Instances of the **ide-disk** class are generic IDE disks with several of tunable parameters. PIO is supported up to mode 4, and Ultra DMA up to level 2.

Attributes

disk_cylinders

Required attribute; **read/write** access; type: `integer`. Number of logical cylinders.

disk_heads

Required attribute; **read/write** access; type: `integer`. Number of logical heads.

disk_sectors

Required attribute; **read/write** access; type: `integer`. Total number of sectors on disk.

disk_sectors_per_track

Required attribute; **read/write** access; type: `integer`. Number of logical sectors per track.

firmware_id

Optional attribute; **read/write** access; type: `string`. Device firmware id. Up to 8 character long string.

geometry_valid

Optional attribute; **read/write** access; type: `integer`. Geometry valid.

image

Required attribute; **read/write** access; type: `object`. Object implementing the image interface holding device data.

model_id

Optional attribute; **read/write** access; type: `string`. Device model id. Up to 40 character long string.

power_mode

Optional attribute; **read/write** access; type: `integer`. Device power mode (0==active, 1==idle, 2==standby, 3==sleep).

stat_atapi_commands

Optional attribute; **read/write** access; type: `[i { 256 }]. (f0, ..., f255)`. Executed ATAPI command count (command *i* has been issued *f_i* times).

stat_bytes_read

Optional attribute; **read/write** access; type: **integer**. Total number of bytes read from device using PIO.

stat_bytes_transferred_dma

Optional attribute; **read/write** access; type: **integer**. Total number of bytes read or written to or from device using DMA.

stat_bytes_written

Optional attribute; **read/write** access; type: **integer**. Total number of bytes written to device using PIO.

stat_commands

Optional attribute; **read/write** access; type: $[i\{256\}] \cdot (f_0, \dots, f_{255})$. Executed command count (command i has been issued f_i times).

Command List

Commands

add-diff-file	add a diff file to the image
add-diff-partial-file	add a partial diff file to the image
add-sun-partition	add partition from a file
create-partition	add a partition to disk
create-sun-vtoc-header	write a new VTOC to a Sun disk
create-sun-vtoc-partition	write partition data in the VTOC on a Sun disk
default-translation	get or set the default CHS translation
delete-sun-vtoc-partition	delete partition data from the VTOC on a Sun disk
dump-sun-partition	write partition as a file
info	print information about the device
print-partition-info	print info about a pc-style partition
print-partition-table	print the disk's pc-style partition table
print-sun-vtoc	print the VTOC for a Sun disk
save-diff-file	save diff file to disk

Command Descriptions

<ide-disk>.add-diff-file

Synopsis

<ide-disk>.add-diff-file *filename* [-replace] [-rw]

Description

Add a diff file to the list of files for an disk. The diff file was typically created with the 'save-diff-file' command, or by a saved configuration. This is basically the same command as <image>.add-diff-file. The file can be made writable instead of read-only using the -rw flag. To replace any existing files, use -replace.

See Also

[`<ide-disk>.save-diff-file`](#), [`<image>.add-diff-file`](#)

`<ide-disk>.add-diff-partial-file`

Synopsis

`<ide-disk>.add-diff-partial-file filename start [size]`

Description

Add a diff file to the list of files for an disk. The diff file was typically created with the 'save-diff-file' command, by one of the dump-*-partition commands, or by a saved configuration. This is basically the same command as `<image>.add-partial-diff-file`.

See Also

[`<ide-disk>.save-diff-file`](#), [`<image>.add-diff-file`](#), [`<image>.add-partial-diff-file`](#)

`<ide-disk>.add-sun-partition`

Synopsis

`<ide-disk>.add-sun-partition number file`

Description

Adds an image or diff as a sun partition to the current disk.

See Also

[`<ide-disk>.dump-sun-partition`](#)

`<ide-disk>.create-partition`

Synopsis

`<ide-disk>.create-partition partition start_cylinder start_head start_sector end_cylinder end_head end_sector (type_id|"type_name") start_lba size_in_sectors [-boot-indication]`

Description

Add a partition to a disk's partition table. Will create a primary PC-style partition with number *partition*. Any existing partition will be overwritten. The start location of the partition is given both in CHS format with *start_cylinder*, *start_head*, *start_sector*, and in linear format with *start_lba*. The partition endpoint is specified in CHS format with *end_cylinder*, *end_head*, and *end_sector*. The size in 512-byte sectors is *size_in_sectors*.

The slight overspecification of the parameters is there because they depend on how the BIOS translates the disk geometry to the BIOS geometry. All CHS coordinates given to this command should be in the translated BIOS geometry.

The partition boot bit is enabled with the *-boot-indication* flag (default off).

The partition type can either raw numerical code, or a symbolic name. The following symbolic names are understood by the command: "unused" (0x0), "DOS 12-bit FAT"

(0x1), "DOS 3.0+ 16-bit FAT" (0x4), "DOS 3.3+ extended" (0x5), "DOS 3.31+ 16-bit FAT" (0x6), "OS/2 IFS" (0x7), "OS/2 boot manager" (0xa), "Win95 32-bit FAT" (0xb), "Win95 32-bit FAT, LBA" (0xc), "Win95 16-bit FAT, LBA" (0xe), "Win95 extended, LBA" (0xf), "Linux swap" (0x82), "Linux native" (0x83), "Linux extended" (0x85), "Linux LVM" (0x8e), "BSD/386" (0xa5), "OpenBSD" (0xa6), "NetBSD" (0xa9), and "BeOS" (0xeb).

See Also

[`<ide-disk>.print-partition-table`](#)

`<ide-disk>.create-sun-vtoc-header`

Synopsis

`<ide-disk>.create-sun-vtoc-header [C] [H] [S] [-quiet]`

Description

Create and write a new VTOC to a Sun disk. The geometry information written is taken from the configuration attribute 'geometry' of the disk, unless specified with the **C**, **H** and **S** parameters. A new empty partition table is also created, with only the standard 'backup' partition as number 2. *-quiet* makes the command silent in case of success.

See Also

[`<ide-disk>.print-sun-vtoc`](#), [`<ide-disk>.create-sun-vtoc-partition`](#), [`<ide-disk>.delete-sun-vtoc-partition`](#)

`<ide-disk>.create-sun-vtoc-partition`

Synopsis

`<ide-disk>.create-sun-vtoc-partition number "tag" "flag" start-block num-blocks [-quiet]`

Description

Write partition information to the VTOC on a Sun disk. This command does not change the format of the disk, and it does not create any file system on the partition. Only the 'Volume Table Of Contents' is modified. No checking is performed to make sure that partitions do not overlap, or that they do not exceed the disk size. *-quiet* makes the command silent in case of success.

See Also

[`<ide-disk>.print-sun-vtoc`](#), [`<ide-disk>.create-sun-vtoc-header`](#), [`<ide-disk>.delete-sun-vtoc-partition`](#)

`<ide-disk>.default-translation`

Synopsis

`<ide-disk>.default-translation [C] [H] [S]`

Description

Set the default CHS translation of a disk, or get the current one if no arguments are given.

<ide-disk>.delete-sun-vtoc-partition

Synopsis

<ide-disk>.delete-sun-vtoc-partition *number* [-quiet]

Description

Delete the information in the VTOC on a Sun disk for the specified partition. No other modification on the disk is performed. -quiet makes the command silent in case of success.

See Also

[**<ide-disk>.print-sun-vtoc**](#), [**<ide-disk>.create-sun-vtoc-header**](#), [**<ide-disk>.create-sun-vtoc-partition**](#)

<ide-disk>.dump-sun-partition

Synopsis

<ide-disk>.dump-sun-partition *number file*

Description

Write all data from a Sun disk partition to the specified file in raw format.

See Also

[**<ide-disk>.print-sun-vtoc**](#), [**<ide-disk>.add-sun-partition**](#)

<ide-disk>.info

Synopsis

<ide-disk>.info

Description

Print detailed information about the configuration of the device.

<ide-disk>.print-partition-info

Synopsis

<ide-disk>.print-partition-info *num [filename]*

Description

Print information about a partition. If a filename is given, then suitable partition parameters for use with image aware e2tools will be written to that file.

<ide-disk>.print-partition-table

Synopsis

<ide-disk>.print-partition-table

Description

Print the PC-style partition table for a disk.

See Also

[`<ide-disk>.create-partition`](#)

<ide-disk>.print-sun-vtoc

Synopsis

<ide-disk>.print-sun-vtoc

Description

Print the contents of the VTOC (volume table of contents) for a Sun disk. This is similar to the Solaris 'prtvtoc' command.

See Also

[`<ide-disk>.create-sun-vtoc-header`](#), [`<ide-disk>.create-sun-vtoc-partition`](#), [`<ide-disk>.delete-sun-vtoc-partition`](#)

<ide-disk>.save-diff-file

Synopsis

<ide-disk>.save-diff-file *filename*

Description

Writes changes to the image as a diff file in craff format. This is basically the same command as <image>.safe-diff-file.

See Also

[`<ide-disk>.add-diff-file`](#), [`<image>.save-diff-file`](#)

image

Provided By

[Simics Core](#)

Interfaces Implemented

[checkpoint](#), [conf_object](#), [image](#), [log_object](#)

Description

The **image** class provides persistent file-backed flat-address storage of devices such as physical memory and disks. Writes are fully shadowed, which allows changes to be stored in a compact way and the original images be kept unchanged. Memory page allocation is done lazily.

In an image object, one or more files represent the image data. This is done in a transparent way: page ranges not present in one file is searched in subsequent files.

Image files can be in raw format, or in compressed CRAFF format. Use the **craff** utility to convert between image file formats.

Attributes

files

Optional attribute; **read/write** access; type: `[[saii]|[saiii]|[saiiii]*]`; **persistent** attribute. `((file, read-only, start, size[, offset]))+` Specifies the files that represent the (initial) image contents. Several files may be specified and may also overlap in memory. Later files override earlier files. *file* is the file name of a file in CRAFF or raw format or a raw disk device. *read-only* is “rw” or “ro” and specifies whether changes to an image should be written back to the file or cached in memory. *start* is the first address in the image represented by the file. *size* is the number of bytes the file represents. If zero, the size is taken from the file. *offset* is the offset in the file where this mapping starts. In checkpoint files generated by Simics, *file* may contain a path of the form `%n%`. Such a sequence will be replaced by the *n*th element of the *checkpoint_path* attribute in the **sim** object, counting from zero. By using `%n%` as path, all absolute paths are kept in the *checkpoint_path* attribute, making it easier to update a checkpoint when files that it depends on have moved.

image_snoop_devices

Optional attribute; **read/write** access; type: `[o*]`. Objects implementing the **image_snoop**. Installed objects will be called through the interface when image data is modified.

init_pattern

Optional attribute; **read/write** access; type: `integer`. Contents of uninitialized bytes (default zero). Must be a single byte.

size

Required attribute; **read/write** access; type: `integer`. Logical image size in bytes.

Command List

Commands

add-diff-file	add a diff file to the image
add-partial-diff-file	add a partial diff file to the image
info	print information about the device
save	save image to disk
save-diff-file	save changes since last checkpoint
set	set bytes in image to specified value
status	print status of the device
x	examine image data

Command Descriptions

`<image>.add-diff-file`

Synopsis

`<image>.add-diff-file filename [-replace] [-rw]`

Description

Adds a diff file to the list of files for an image. The diff file was typically created with **save-diff-file**, or by a saved configuration. The file can be made writable instead of read-only using the `-rw` flag. To replace any existing files, use `-replace`. This command should not be issued if the image contains changed (unsaved) data.

See Also

[`<image>.save-diff-file`](#)

`<image>.add-partial-diff-file`

Synopsis

`<image>.add-partial-diff-file filename start [size]`

Description

Adds a partial diff file to the list of files for an image. The diff file was typically created with the 'save-diff-file' command, by one of the dump-* partition commands, or by a saved configuration.

See Also

[`<image>.add-diff-file`](#), [`<image>.save-diff-file`](#)

`<image>.info`

Synopsis

`<image>.info`

Description

Print detailed information about the configuration of the device.

<image>.save

Synopsis

<image>.save *filename* [*start-byte*] [*length*]

Description

Writes the image binary data (in raw form) to *filename*. If *start* and/or *length* are given, they specify the start offset and number of bytes to write respectively; otherwise, the whole image is copied.

<image>.save-diff-file

Synopsis

<image>.save-diff-file *filename*

Description

Writes changes to the image since the last checkpoint to a named file in craff format.

See Also

[<image>.add-diff-file](#), [<image>.save](#)

<image>.set

Synopsis

<image>.set *address* *value* [*size*] [-l] [-b]

Description

Sets *size* bytes in an image at offset *address* to *value*. The default *size* is 4 bytes, but can be anywhere between 1 and 8 inclusive.

If *value* is larger than the specified size, behavior is undefined.

The -l and -b flags are used to select little-endian and big-endian byte order, respectively. If neither is given, the byte order of the currently selected processor is used.

See Also

[set](#)

<image>.status

Synopsis

<image>.status

Description

Print detailed information about the current status of the device.

<image>.x

Synopsis

`<image>.x offset [size] [-c]`

Description

Displays *length* bytes starting at *offset* from the image. The *-c* flag compresses the output by not displaying sequences of zeros.

ISP1040

Provided By
[ISP1040](#)

Aliases

ISP1040_SUN

Interfaces Implemented

`conf_object, io_memory, log_object`

Description

The ISP1040 SCSI Controller.

Attributes

scsi_bus

Required attribute; **read/write** access; type: `object`. The name of the SCSI bus that the device is connected to. This object must implement the 'scsi-bus' interface.

scsi_id

Required attribute; **read/write** access; type: `integer`. The (target) id on the scsi bus for the controller itself.

Command List

Commands

<code>info</code>	print information about the device
<code>pci-header</code>	<i>deprecated</i> — print PCI device header
<code>print-pci-config-reg</code>	print PCI configuration registers
<code>status</code>	print status of the device

Command Descriptions

`<ISP1040>.info`

Synopsis

`<ISP1040>.info`

Description

Print detailed information about the configuration of the device.

`<ISP1040>.pci-header — deprecated`

Synopsis

`<ISP1040>.pci-header [-v]`

Description

This command is deprecated; use `<ISP1040>.print-pci-config-reg`s instead.

```
<ISP1040>.print-pci-config-reg
```

Synopsis

```
<ISP1040>.print-pci-config-reg [-v]
```

Description

Print the PCI configuration registers. The `-v` flag turns on verbose mode.

```
<ISP1040>.status
```

Synopsis

```
<ISP1040>.status
```

Description

Print detailed information about the current status of the device.

ISP2200

Provided By
[ISP2200](#)

Aliases

ISP2200_SUN

Interfaces Implemented

[conf_object](#), [io_memory](#), [log_object](#)

Description

The ISP2200 Fibre Channel SCSI Controller.

Attributes

loop_id

Required attribute; **read/write** access; type: `integer`. The loop id of the controller itself, on the FC-AL

Command List

Commands

info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
status	print status of the device

Command Descriptions

`<ISP2200>.info`

Synopsis

`<ISP2200>.info`

Description

Print detailed information about the configuration of the device.

`<ISP2200>.pci-header — deprecated`

Synopsis

`<ISP2200>.pci-header [-v]`

Description

This command is deprecated; use `<ISP2200>.print-pci-config-reg`s instead.

`<ISP2200>.print-pci-config-reg`

Synopsis

```
<ISP2200>.print-pci-config-reg [-v]
```

Description

Print the PCI configuration registers. The *-v* flag turns on verbose mode.

```
<ISP2200>.status
```

Synopsis

```
<ISP2200>.status
```

Description

Print detailed information about the current status of the device.

mac_splitter

Provided By
[mac-splitter](#)

Interfaces Implemented
`conf_object`, [ieee_802_3_mac_v3](#), [log_object](#)

Description

This object splits incoming access from one PHY to several MACs.

Attributes

mac

Required attribute; **read/write** access; type: `[o|[os]{2}]`. The MACs connected to the splitter.

mem-traffic-meter

Provided By
[mem-traffic-meter](#)

Interfaces Implemented
conf_object, [log_object](#), [snoop_memory](#)

Ports

source_byte (scalar_time), source_transaction (scalar_time), target_byte (scalar_time),
target_transaction (scalar_time)

Description

A **mem-traffic-meter** object snoops on the memory traffic passing through one or more **memory-space** objects, and makes the data available to the graphical plot window.

Command List

Commands
[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

<mem-traffic-meter>.info

Synopsis
<mem-traffic-meter>.info

Description

Print detailed information about the configuration of the device.

<mem-traffic-meter>.status

Synopsis
<mem-traffic-meter>.status

Description

Print detailed information about the current status of the device.

memory-space

Provided By

[Simics Core](#)

Interfaces Implemented

[breakpoint](#), [conf_object](#), [log_object](#), [map_demap](#), [memory_space](#), [translate](#)

Description

Instances of the **memory-space** class provide linear address spaces into which devices and memory can be mapped.

Attributes

default_target

Optional attribute; **read/write** access; type: $[o|[os|n]iio|[os|n]|n]$ or **nil**. (*object, function, offset, target*). An access not targeting any object in the map attribute will be forwarded to the default target. *target* should be non-nil if and only if *object* is a translation object.

map

Optional attribute; **read/write** access; type: $[[io|[os|n]iii]|[io|[os|n]iiio|[os|n]|n]|[io|[os|n]iiio|[os|n]|ni]| [io|[os|n]iiio|[os|n]|nii]| [io|[os|n]iiio|[os|n]|niii]*]$. ((*base, object* **or** (*object, "port*"), *function, offset, length, target, priority, align-size, byte-swap*)*).

Maps zero or more objects in the memory space. **object** must implement one of the **translate**, **bridge**, **ram**, **io_memory**, **port_space**, or **rom** interfaces, searched for in that order. (Some of these interfaces are Simics internal and can not be implemented by user-defined classes).

If **object** implements the **translate** or **bridge** interface, then the **target** field must be set to an object, which in turn must implement either the **memory_space**, **ram** or **rom** interface, **target** should be **NIL** otherwise.

object is mapped in the address space from *base* to *base + length - 1* (inclusive). Both *function* and *offset* are handed on to the mapped **object** through the appropriate interface. The function number, that is device specific, provides a way for devices with multiple mappings to figure out which mapping that is accessed. Starting with Simics 3.2, *function* has been superseeded by the (*object, "port*") syntax, which maps a specific port (i.e. bank) of the **object**. The *priority* (-32768 – 32767) is used when multiple mappings exist on the same address. The mapping with the lowest number has the highest priority.

If the map target does not support large accesses, then *align-size* can be set to the maximum allowed size (which must be a power of two). Accesses spanning align boundaries will be split into several smaller transactions. By default, the align size

is set to 8 for devices and 8192 for memory. Mappings with an *align-size* of 2, 4, or 8 may set the *byte-swap* field to 1. This can be used to model bridges that perform byte swapping on a specific bus width. It is also possible to set *byte-swap* to 3 to swap all bytes in the transaction based on the access size. The bus and transaction byte-swap variants can be combined by setting a value of 2. Consider a mapping with a 4-byte align size and memory at address 0 that contains the bytes: 0x00 0x01 0x02 0x03. A 2-byte big-endian read access at address 0 will give the following values as results. No swap: 0x0001, bus swap: 0x0302, bus and transaction swap: 0x0203 and transaction swap: 0x0100

The most commonly used syntax for this attribute is:

```
memory-space.map = [  
    [0xa000, [obj], 'foo'], 0, 0, 0x1000],  
    [0xf000, [obj], 'bar'], 0, 0, 0x0200]  
]
```

This will map the bank *foo* at 0xa000. The length of the mapping is 0x1000 bytes. Similary, the bank *bar* will be mapped at 0xf000 with a length of 0x0200 bytes.

memory

Pseudo attribute; **read/write** access; type: `data`; **integer** or **list** indexed; indexed type: `[i*]`, `data`, or `integer`. Read from or write to the space. Data is read without side effects (with the inquiry bit set), while writes are allowed to trigger side effects (including actually writing the data).

snoop_device

Optional attribute; **read/write** access; type: `object` or `nil`. The *snoop_device* is similar to a timing-model, but called after a memory operation is complete. The object has to implement the “*snoop_memory*” interface, but the returned stall time is ignored.

timing_model

Optional attribute; **read/write** access; type: `object` or `nil`. The *timing_model* is called when a memory operation reaches the *memory-space*, before it is performed. The model has to implement the “*timing_model*” interface.

Command List

Commands defined by interface **breakpoint**

[break](#), [tbreak](#)

Commands defined by interface **memory_space**

[debug](#)

Commands

[add-map](#) map device in a memory-space

del-map	remove device map from a memory-space
get	get value of physical address without side-effects
info	print information about the device
load-binary	load binary (executable) file into memory
load-file	load file into memory
map	list memory map
read	get value of physical address
set	set physical address to specified value without side-effects
status	print status of the device
write	set physical address to specified value
x	examine raw memory contents

Command Descriptions

`<memory-space>.add-map`

Synopsis

```
<memory-space>.add-map device base length [function] [offset] [target] [priority] [align-size] ["swap"]
```

Description

Map *device* into a memory-space at address *base* and with length *length*. The *device* parameter can be *device:port* to map a specific port of the device. Otherwise, mappings of the same device may be identified by a device-specific *function* number.

For translator and bridge mappings, a *target* device should be given.

The mapping may specify an offset into the device's memory space, using the *offset* argument. If several device mappings overlap, the *priority* is used to select what device will receive memory accesses. The priority is an integer between -32768 and 32767; lower numbers have higher priority.

For devices that do not support large accesses, the *align-size* governs how accesses are split before the device is accessed. A device mapping may swap the bytes of an access based on the *swap* argument, that should be one of none, bus, bus-trans and trans. For a description of these, see the documentation of the *map* attribute in the **memory-space** class.

See Also

[`<memory-space>.map`](#), [`<memory-space>.del-map`](#)

`<memory-space>.del-map`

Synopsis

```
<memory-space>.del-map device [function] [base]
```

Description

Remove the mapping of *device* from a memory-space.

The *device* parameter can be *device:port* to remove the mapping of a specific port.

If a function number is given by the *function* argument, then only mappings with a matching number are removed.

If *base* address is specified, only mappings with the matching address are removed.

If both *function* and *base* address are specified, only mappings with a matching function number, at the specified address, are removed.

When used in an expression, the command returns the number of removed mappings.

See Also

[`<memory-space>.map`](#), [`<memory-space>.add-map`](#)

`<memory-space>.get`

Synopsis

```
<memory-space>.get address [size] [-l] [-b]
<memory-space>.read address [size] [-l] [-b]
<port-space>.get address [size] [-l] [-b]
<port-space>.read address [size] [-l] [-b]
get address [size] [-l] [-b]
```

Description

Get value of physical memory location. The *size* argument specifies how many bytes should be read. This defaults to 4, but can be any number of bytes between 1 and 8 (inclusive) for memory-spaces and between 1 and 8 (inclusive) for port-spaces.

The *-l* and *-b* flags are used to select little-endian and big-endian byte order, respectively. If neither is given, the byte order of the currently selected processor is used.

The **get** command performs the access in inquiry mode without triggering any side-effects while **read** may trigger side-effects.

The non-namespace version of this command operates on the physical memory associated with the current processor.

See Also

[`x`](#), [`set`](#), [`signed`](#)

`<memory-space>.info`

Synopsis

```
<memory-space>.info
```

Description

Print detailed information about the configuration of the device.

`<memory-space>.load-binary`

Synopsis

```
<memory-space>.load-binary filename [offset] [-v] [-pa] [-n]
load-binary filename [offset] [-v] [-pa] [-l] [-n]
```

Description

Load a binary (executable) file into the given physical or virtual memory space. The supported formats are ELF, Motorola S-Record, PE32 and PE32+.

By default the virtual load address from the file is used. The physical load address can be used instead, for file formats supporting both, by specifying the `-pa` flag. The load address selected does not affect if the binary is loaded into the virtual or physical address space.

If an `offset` is supplied, it will be added to the load address taken from the file.

The global **load-binary** command will use the currently selected processor to find the memory space to load the binary into. If the `-l` flag is given, it will load it into the virtual memory space, otherwise it will use the physical memory space. The processor must have a valid virtual to physical translation set up.

When using the namespace command on a **processor** object, it will load the binary into the virtual memory space of that processor.

When using the namespace command on a **memory-space** object, it will load the binary directly into that memory space without any virtual to physical translation.

The `-v` flag turns on verbose mode, printing information about the loaded file.

The `-n` flags tells the command to not clear `.bss` areas in the file.

The return value is the address of the execution entry point. This value is typically used in a call to **set-pc**.

load-binary uses Simics's Search Path and path markers (%simics%, %script%) to find the file to load. Refer to *The Command Line Interface* chapter of the *Hindsight User's Guide* manual for more information on how Simics's Search Path is used to locate files.

See Also

[load-file](#), [add-directory](#)

<memory-space>.load-file

Synopsis

```
<memory-space>.load-file filename [offset]
load-file filename [offset]
```

Description

Loads the contents of the file named `filename` into memory (defaulting to the current frontend processor's physical memory space), starting at physical address `offset`. Default offset is 0.

load-file uses Simics's Search Path and path markers (%simics%, %script%) to find the file to load. Refer to *The Command Line Interface* chapter of the *Hindsight User's Guide* manual for more information on how Simics's Search Path is used to locate files.

See Also

[load-binary](#), [add-directory](#)

<memory-space>.map

Synopsis

<memory-space>.map

Description

Prints the memory map of the memory space object, one line per entry in the map attribute of the memory space. The *base* column is the starting address of the map. The *object* column contains the object mapped at that address. *fn* is the function number and *offs* is the offset for the object. *length* is the number of bytes mapped.

See Also

[<memory-space>.add-map](#), [<memory-space>.del-map](#)

<memory-space>.read

Synopsis

```
<memory-space>.read address [size] [-l] [-b]
<memory-space>.get address [size] [-l] [-b]
<port-space>.get address [size] [-l] [-b]
<port-space>.read address [size] [-l] [-b]
get address [size] [-l] [-b]
```

Description

Get value of physical memory location. The size argument specifies how many bytes should be read. This defaults to 4, but can be any number of bytes between 1 and 8 (inclusive) for memory-spaces and between 1 and 8 (inclusive) for port-spaces.

The *-l* and *-b* flags are used to select little-endian and big-endian byte order, respectively. If neither is given, the byte order of the currently selected processor is used.

The **get** command performs the access in inquiry mode without triggering any side-effects while **read** may trigger side-effects.

The non-namespace version of this command operates on the physical memory associated with the current processor.

See Also

[x](#), [set](#), [signed](#)

<memory-space>.set

Synopsis

```
<memory-space>.set address value [size] [-l] [-b]
<memory-space>.write address value [size] [-l] [-b]
<port-space>.set address value [size] [-l] [-b]
<port-space>.write address value [size] [-l] [-b]
set address value [size] [-l] [-b]
```

Description

Set the *size* bytes of physical memory at location *address* to *value*. The default *size* is 4 bytes, but can be anywhere between 1 and 8 (inclusive) for memory-spaces and between 1 and 4 (inclusive) for port-spaces.

If *value* is larger than the specified size, behavior is undefined.

The *-l* and *-b* flags are used to select little-endian and big-endian byte order, respectively. If neither is given, the byte order of the currently selected processor is used.

The **set** command performs the access in inquiry mode without triggering any side-effects while **write** may trigger side-effects.

If the *value* argument is a list, then each item is written to memory as a value of *size* bytes, starting at *address*. The byte order flags operate on each item in the list.

The non-namespace version of this command operates on the physical memory associated with the current processor.

See Also

[get](#), [x](#), [pselect](#)

<memory-space>.status

Synopsis

```
<memory-space>.status
```

Description

Print detailed information about the current status of the device.

<memory-space>.write

Synopsis

```
<memory-space>.write address value [size] [-l] [-b]
<memory-space>.set address value [size] [-l] [-b]
<port-space>.set address value [size] [-l] [-b]
<port-space>.write address value [size] [-l] [-b]
set address value [size] [-l] [-b]
```

Description

Set the *size* bytes of physical memory at location *address* to *value*. The default *size* is

4 bytes, but can be anywhere between 1 and 8 (inclusive) for memory-spaces and between 1 and 4 (inclusive) for port-spaces.

If *value* is larger than the specified size, behavior is undefined.

The **-l** and **-b** flags are used to select little-endian and big-endian byte order, respectively. If neither is given, the byte order of the currently selected processor is used.

The **set** command performs the access in inquiry mode without triggering any side-effects while **write** may trigger side-effects.

If the *value* argument is a list, then each item is written to memory as a value of *size* bytes, starting at *address*. The byte order flags operate on each item in the list.

The non-namespace version of this command operates on the physical memory associated with the current processor.

See Also

[get](#), [x](#), [pselect](#)

<memory-space>.x

Synopsis

```
<memory-space>.x address [size] [-c]
x [cpu-name] address [size] [-c]
```

Description

Display the contents of a memory space starting at *address*. Either the memory space is explicitly specified as in **<memory-space>.x** or the CPU connected to the memory space can be specified; e.g., **<processor>.x**. By itself, **x** operates on the memory connected to the current frontend processor.

If the memory is accessed via a CPU, the type of *address* is specified by a prefix. For physical addresses use **p:address**; for virtual addresses, **v:address** on non-x86 targets. On x86, use **segment-register:offset** or **l:address** for x86 linear addresses.

If no prefix is given it will be interpreted as a virtual address. On x86 the default is **ds:address** (data segment addressing).

The access will be made in inquiry mode, which means it will have no side-effects on the CPU or the accessed object. Use the **<memory-space>.read** command to do non-inquiry accesses.

The *size* argument specifies the number of bytes to examine. When examining virtual memory, only addresses which can be found in the TLB or hardware page tables (if any) are shown. Unmapped addresses are shown as “--”, undefined physical addresses as “**”.

The **-c** flag compresses the output by not displaying sequences of zeros.

See Also

[disassemble](#), [get](#), [set](#)

microwire-eeprom

Provided By

[microwire-eeprom](#)

Interfaces Implemented

`conf_object`, [log_object](#), [microwire](#)

Description

This device models a generic, 3-wire (microwire) serial EEPROM (for example 93Cxx).

mii-management-bus

Provided By
[mii-management-bus](#)

Interfaces Implemented
`conf_object`, [log_object](#), [mdio45_bus](#), [mii_management](#)

Ports
MDC ([signal](#)), MDIO ([signal](#))

Description

The mii-management-bus device is a bus model that allows multiple MII devices to share the same management interface. The interface is usually a two pin (MDC/MDIO) serial connection, and all devices are assumed to implement the “mii_management” interface.

Attributes

devices

Optional attribute; **read/write** access; type: `[[oi]*]`. A list of device and address pairs, listing MII capable devices connected to the this management bus, i.e., sharing the same MDC and MDIO pins. Note that all these devices must implement the “mii_management” interface.

Command List

Commands

[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

`<mii-management-bus>.info`

Synopsis

`<mii-management-bus>.info`

Description

Print detailed information about the configuration of the device.

`<mii-management-bus>.status`

Synopsis

`<mii-management-bus>.status`

Description

Print detailed information about the current status of the device.

mii-transceiver

Provided By

[mii-transceiver](#)

Interfaces Implemented

[conf_object](#), [ethernet_common](#), [ethernet_device](#), [ieee_802_3_phy](#), [ieee_802_3_phy_v2](#), [ieee_802_3_phy_v3](#), [log_object](#), [mii](#), [mii_management](#)

Ports

MDC ([signal](#)), MDIO ([signal](#))

Description

mii-transceiver is an IEEE 802.3 physical layer device with MII interface

Attributes

add_crc_on_inject

Optional attribute; **read/write** access; type: `integer`. Frames injected using the ‘inject_packet’ will get a correctly calculated CRC added at the end when this attribute is set to 1 (default). When set to 0, the user has to supply a CRC field with the injected frame. Note that you must always provide room for the CRC field, even when this attribute is set to 1.

address

Optional attribute; **read/write** access; type: `integer`. PHY identifier sent in calls to `ieee_802_3_mac` interface methods.

duplex_mode

Optional attribute; **read/write** access; type: `integer`. Set to 1 for full duplex (0 for half duplex).

inject_packet

Pseudo attribute; **write-only** access; type: `data`. Attribute used to send a packet to the network device. Writing this attribute at any time injects a new packet into the device (without involving the network simulation). Injecting a packet copies the packet data, allowing the caller to reuse or dispose of the buffer used for creating the packet, after the attribute is written.

last_frame

Pseudo attribute; **read/write** access; type: `data` or `nil`. The frame that is currently about to be sent or received. The format is a raw Ethernet frame. This attribute is only valid in `Ethernet_Transmit` and `Ethernet_Receive` hap callbacks. It is possible to override the packet by assigning this attribute. The device will drop the current packet if the attribute is set to `Nil`, or with data of zero length.

link

Optional attribute; **read/write** access; type: `object` or `nil`. The Ethernet link that the network device is connected to.

link_speed

Optional attribute; **read/write** access; type: `integer`. The link speed (usually 10 or 100 Mbit/s).

link_status

Optional attribute; **read/write** access; type: `integer`. The status of the link - 0 when completely unconnected, 1 when connected but down, and 2 when connected and up.

mac

Required attribute; **read/write** access; type: `[os] or object`. Set to a media access controller (MAC) object, implementing the `ieee_802_3_mac_v3` or `ieee_802_3_mac` interface.

mac_address

Optional attribute; **read/write** access; type: `[i{6}]`, `string`, or `nil`. Ethernet (MAC) address of the network interface.

model_crc

Optional attribute; **read/write** access; type: `integer`. If set to 1, the device will calculate and include an Ethernet CRC on all transmitted frames, and check the CRC on received frames. This attribute is **ignored** when new-style links are used.

registers

Optional attribute; **read/write** access; type: `[i{32}]`. The MII management register set as described in IEEE 802.3.

tx_bandwidth

Optional attribute; **read/write** access; type: `integer`. The transmit bandwidth of the network interface in bits per second. The network interface will limit the rate at which it sends packets to remain below this bandwidth. Set to 0 for unlimited bandwidth.

Command List

Commands

<code>connect</code>	<i>deprecated</i> — connect to a simulated Ethernet link
<code>disconnect</code>	<i>deprecated</i> — disconnect from simulated link
<code>info</code>	print information about the device
<code>status</code>	print status of the device

Command Descriptions

`<mii-transceiver>.connect — deprecated`

Synopsis

`<mii-transceiver>.connect [-auto] link`

Description

This command is deprecated; use [connect](#) instead.

Connect the device to a simulated Ethernet link. The flag '-auto' is deprecated and shouldn't be used.

See Also

[`<mii-transceiver>.disconnect`](#)

[`<mii-transceiver>.disconnect — deprecated`](#)

Synopsis

[`<mii-transceiver>.disconnect`](#)

Description

This command is deprecated; use [disconnect](#) instead.

Disconnect the device from a simulated Ethernet link.

See Also

[`<mii-transceiver>.connect`](#)

[`<mii-transceiver>.info`](#)

Synopsis

[`<mii-transceiver>.info`](#)

Description

Print detailed information about the configuration of the device.

[`<mii-transceiver>.status`](#)

Synopsis

[`<mii-transceiver>.status`](#)

Description

Print detailed information about the current status of the device.

ms1553-link

Provided By
[ms1553-link](#)

Interfaces Implemented
conf_object, [log_object](#), [ms1553_link](#)

Description

The **ms1553-link** models a MIL-STD-1553 bus, called link in Simics. It provides interfaces to bus controllers, remote terminal devices and bus monitors. The link keeps track of the protocol state to aid debugging of 1553 device and driver development. It also supports error injection and data inspection.

Command List

Commands

capture-start	start traffic recorder
capture-stop	stop traffic recorder
info	print information about the device
playback-start	start traffic generator
playback-stop	stop traffic generation
status	print status of the device

Command Descriptions

<ms1553-link>.capture-start

Synopsis

[<ms1553-link>.capture-start](#) *filename*

Description

Starts recording bus traffic to a specified file, in a format that can be played back. If the file exist, it is silently overwritten.

See Also

[<ms1553-link>.capture-stop](#)

<ms1553-link>.capture-stop

Synopsis

[<ms1553-link>.capture-stop](#)

Description

Stop recording bus traffic previously started with start-recorder.

See Also

[<ms1553-link>.playback-start](#)

<ms1553-link>.info

Synopsis

<ms1553-link>.info

Description

Print detailed information about the configuration of the device.

<ms1553-link>.playback-start

Synopsis

<ms1553-link>.playback-start *filename*

Description

Starts generating bus traffic from a specified file. The file should be of PASS-3200 dump format.

See Also

[**<ms1553-link>.capture-start**](#), [**<ms1553-link>.playback-stop**](#)

<ms1553-link>.playback-stop

Synopsis

<ms1553-link>.playback-stop

Description

Stop bus traffic generation previously started with playback-start.

See Also

[**<ms1553-link>.playback-start**](#)

<ms1553-link>.status

Synopsis

<ms1553-link>.status

Description

Print detailed information about the current status of the device.

ms1553-recorder-bm

Provided By

[ms1553-link](#)

Interfaces Implemented

`conf_object`, [log_object](#), [ms1553_terminal](#)

Description

Fake MIL-STD-1553 Bus Monitor used for recording.

ms1553-test-rt

Provided By
[ms1553-test-rt](#)

Interfaces Implemented
conf_object, [log_object](#), [ms1553_terminal](#)

Description

Simple MIL-STD-1553 Remote Terminal used for testing. This RT can be configured with a number of sub-addresses using the *sub_addresses* attribute. For each sub-address, a static list of 16-bit words should be supplied that specifies what the RT will return on “Transmit” requests. The data may be 1 up to 32 words in length. For “Receive” requests, the RT will report the received words using Simics standard log functions.

Attributes

sub_addresses

Required attribute; **read/write** access; type: `[[i [i*]]*]`. A list of sub-addresses that the RT will implement, and for each sub-address there is a non-empty list of up to 32 16-bit words that are returned on reads from that sub-address.

terminal_address

Required attribute; **read/write** access; type: `integer`. The Terminal Address of the RT.

ms1553_rt

Provided By
[ms1553-rt](#)

Interfaces Implemented

`conf_object`, [log_object](#), [ms1553_bridge_terminal](#), [ms1553_terminal](#)

Description

This **ms1553_rt** is an example model of a remote terminal (RT) according to the MIL-STD-1553 standard. The RT can either connect to a virtual ms1553-bus or real ms1553-bus. The *link* connector connects the RT to a virtual bus. The *bridge* connector connects the RT to a real bus via a *bridge*. The *bridge* controls a host card that emulates the RT. The emulated RT is referred to as the shadow RT, as it is identical to the RT in the Simics world. It is up to the RT in Simics to keep the shadow RT in sync with itself.

Attributes

disconnect_bridge

Pseudo attribute; **read/write** access; type: `integer`. Disconnect from bridge.

last_command

Optional attribute; **read/write** access; type: `integer`. The last command word.

status_word

Optional attribute; **read/write** access; type: `integer`. The status word of the RT.

sub_addresses

Required attribute; **read/write** access; type: `[[i [i*] [i*]] *]`. The Terminal Address of the RT.

terminal_address

Required attribute; **read/write** access; type: `integer`. The terminal address of the RT.

mtprof

Provided By
mtprof

Interfaces Implemented
conf_object, log_object

Description

The **mtprof** class is used for multithreaded performance profiling.

Command List

Commands

cellstat	display cell profiling information
disable	disable multithreaded simulation profiling
enable	enable multithreaded simulation profiling
info	print information about the device
modelstat	display ideal execution time on a sufficiently parallel host
save-data	save profiling data to file
status	print status of the device

Command Descriptions

<mtprof>.cellstat

Synopsis

<mtprof>.cellstat

Description

The **cellstat** command present a table with per-cell information about how much cpu-time has been spent simulating processors and devices.

The *rt* column contains the accumulated time measured in seconds. The *%elapsed* column contains the same data expressed as a fraction of the time the simulation has been running. The *%cpu time* column, presents the data as a fraction of the total cpu time used for the simulation.

<mtprof>.disable

Synopsis

<mtprof>.disable
<mtprof>.enable [*interval*]

Description

Enable multithreaded simulation profiling. The amount of host cpu time required to simulate each cell is measured every *interval* virtual ms.

The collected data is fed into a performance model which estimates how fast the simulation would run on a system with enough host cores and Simics Accelerator licenses to allow each cell to run on a dedicated core. The performance model also gives some insights into the performance implications of various min-latency settings (settable though the **set-min-latency** commands).

The *interval* parameter should normally be set to a value of the same order as the min-latency setting of interest.

<mtprof>.enable

Synopsis

```
<mtprof>.enable [interval]  
<mtprof>.disable
```

Description

Enable multithreaded simulation profiling. The amount of host cpu time required to simulate each cell is measured every *interval* virtual ms.

The collected data is fed into a performance model which estimates how fast the simulation would run on a system with enough host cores and Simics Accelerator licenses to allow each cell to run on a dedicated core. The performance model also gives some insights into the performance implications of various min-latency settings (settable though the **set-min-latency** commands).

The *interval* parameter should normally be set to a value of the same order as the min-latency setting of interest.

<mtprof>.info

Synopsis

```
<mtprof>.info
```

Description

Print detailed information about the configuration of the device.

<mtprof>.modelstat

Synopsis

```
<mtprof>.modelstat
```

Description

The **modelstat** command can be used to estimate how fast the simulation would run on a host machine with enough cores and Simics Accelerator licenses to allow each cell to run on a dedicated host thread.

The *latency* column roughly corresponds to different min-latency settings (see **set-min-latency**).

The performance model which generates the estimate essentially models the time synchronization mechanism which keeps the virtual time in different cells in sync. It is important to note that the model does not factor in target behavior changes due to a deffering latency setting, and this is often a major factor.

The lowest latency to be modeled can be specified with the **enable-mtprof**.

<mtprof>.save-data

Synopsis

```
<mtprof>.save-data [file] [-model] [-no-cell-data] [-octave] [-oplot]
```

Description

Save collected cell performance data to *file* in the form of a table. The default columns are virtual time, real time and per-cell cpu time.

Predicted execution time on a sufficiently parallel host machine is included in the table if the command is *-model* switch is used.

The *-octave* or *-oplot* switches can be used to generate output data suitable for Octave. The later of these two options causes explicit plot commands to be included in the output.

The *-no-cell-data* switch causes per-cell data to be omitted from the table.

The output data is either printed on stdout or saved the *file*, if specified.

<mtprof>.status

Synopsis

```
<mtprof>.status
```

Description

Print detailed information about the current status of the device.

NS16450

Provided By
[NS16x50](#)

Interfaces Implemented

`conf_object`, [io_memory](#), [log_object](#), [serial_device](#)

Ports

HRESET ([signal](#)), Reset ([signal](#)), SRESET ([signal](#)), regs ([int_register](#), [io_memory](#))

Description

UART (Universal Asynchronous Receiver Transmitter) is a popular method of serial asynchronous communication. Typically, the UART is connected between a processor and a peripheral. To the processor, the UART appears as an 8-bit read-write parallel port that performs serial-to-parallel conversions for the processor, and vice versa for the peripheral.

Attributes

interrupt_mask_out2

Optional attribute; **read/write** access; type: `integer`. If set to non-zero, then the OUT2 pin is used to mask the output interrupt pin, meaning that no interrupt will be raised unless OUT2 is high.

receive_ready_waiting

Optional attribute; **read/write** access; type: `boolean`. True when data has started to drop because the connected device returned that it couldn't receive the data during last write cycle.

receive_throttled

Optional attribute; **read/write** access; type: `boolean`. True when the incoming data is throttled because it handles overruns safely rather than strict.

Command List

Commands

`info` print information about the device
`status` print status of the device

Command Descriptions

<NS16450>.info

Synopsis

<NS16450>.info

Description

Print detailed information about the configuration of the device.

```
<NS16450>.status
```

Synopsis

```
<NS16450>.status
```

Description

Print detailed information about the current status of the device.

NS16550

Provided By
[NS16x50](#)

Interfaces Implemented

`conf_object`, [io_memory](#), `log_object`, `serial_device`

Ports

HRESET ([signal](#)), Reset ([signal](#)), SRESET ([signal](#)), regs ([int_register](#), [io_memory](#))

Description

UART (Universal Asynchronous Receiver Transmitter) is a popular method of serial asynchronous communication. Typically, the UART is connected between a processor and a peripheral. To the processor, the UART appears as an 8-bit read-write parallel port that performs serial-to-parallel conversions for the processor, and vice versa for the peripheral.

Attributes

interrupt_mask_out2

Optional attribute; **read/write** access; type: `integer`. If set to non-zero, then the OUT2 pin is used to mask the output interrupt pin, meaning that no interrupt will be raised unless OUT2 is high.

receive_ready_waiting

Optional attribute; **read/write** access; type: `boolean`. True when data has started to drop because the connected device returned that it couldn't receive the data during last write cycle.

receive_throttled

Optional attribute; **read/write** access; type: `boolean`. True when the incoming data is throttled because it handles overruns safely rather than strict.

Command List

Commands

`info` print information about the device
`status` print status of the device

Command Descriptions

<NS16550>.info

Synopsis

<NS16550>.info

Description

Print detailed information about the configuration of the device.

```
<NS16550>.status
```

Synopsis

```
<NS16550>.status
```

Description

Print detailed information about the current status of the device.

NS16550_c

Provided By
[NS16550_c](#)

Interfaces Implemented

`conf_object`, `io_memory`, `log_object`, `rs232_device`, `serial_device`

Ports

Reset ([signal](#))

Description

National Semiconductor's NS16550, a Universal Asynchronous Receiver/Transmitter (UART).

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<NS16550_c>.info

Synopsis

`<NS16550_c>.info`

Description

Print detailed information about the configuration of the device.

<NS16550_c>.status

Synopsis

`<NS16550_c>.status`

Description

Print detailed information about the current status of the device.

onfi_flash

Provided By
[onfi-flash](#)

Interfaces Implemented

`conf_object, io_memory, log_object, nand_flash`

Ports

`regs (int_register, io_memory)`

Description

The **onfi_flash** class is a generic NAND Flash Memory model that is compliant to ONFI 1.0 standard.

The following commands are supported:

Read

Change Read Column

Block Erase

Read Status

Read Status Enhanced

Page Program

Change Write Column

Read ID

Read Parameter Page

Reset

Other commands are currently not supported.

The **data_image** and **spare_image** attributes must points to image objects holding the data and spare area data of the flash of size consistent with the parameter of the device.

The following attributes regarding device parameters are required to be properly set before object instantiation (and can not be changed after configuration):

bus_width: data bus width of this device, either 8 or 16

data_bytes_per_page: either 512 or 2048

pages_per_block: 16/32, or 64, depending on **data_bytes_per_page**

blocks: total number of blocks, must be power of 2

id_bytes: the electronic signature of this device returned by **Read ID**

parameter_page: 256+ bytes of device parameters read by **Read Parameter Page**

The following attributes can be read for information purpose:

spare_bytes_per_page: 16 or 64 depending on **data_bytes_per_page**

total_bytes_per_page: 528 or 2112 depending on **data_bytes_per_page**

Further information can be found in the documentations of the individual attributes.

Notes

Devices with 16-bit bus width will interpret the data in the images as little-endian. For example, if column 0 of page 0 is read, byte 0 in the image will be returned in bit 0 - 7 and byte 1 in bit 8-15. If you use an image in big-endian format you must therefore byte-swap it before loading it.

Attributes

address_latch_enable

Optional attribute; **read/write** access; type: `integer`. The current state of the “address latch enable” pin, true (non-zero) for high and false (zero) for low.

blocks

Required attribute; **read/write** access; type: `integer`. The number of blocks of the device. Must be a power of 2.

bus_width

Required attribute; **read/write** access; type: `integer`. Data bus width of the device. Must be either 8 or 16.

command_latch_enable

Optional attribute; **read/write** access; type: `integer`. The current state of the “command latch enable” pin, true (non-zero) for high and false (zero) for low.

data_bytes_per_page

Required attribute; **read/write** access; type: `integer`. The number of data bytes per page. Must be 512 or 2048.

data_image

Required attribute; **read/write** access; type: `[os] or object`. The image object containing the data part of the device’s pages. The image should contain the data contents of the pages concatenated in order.

id_bytes

Required attribute; **read/write** access; type: `[i*]`. Bytes returned by the “read ID (0x00)” command.

pages_per_block

Required attribute; **read/write** access; type: `integer`. The number of pages per block. Must be 16 or 32 for devices with 512-byte pages and 64 for devices with 2048-byte pages.

parameter_page

Required attribute; **read/write** access; type: `[i*]`. bytes returned by the “Read Parameter Page” command.

saved_state

Optional attribute; **read/write** access; type: `integer`. Saved state.

saved_state_counter

Optional attribute; **read/write** access; type: integer. Saved sub-state.

spare_bytes_per_page

Pseudo attribute; **read/write** access; type: integer. The number of spare bytes per page (Read Only). 16 for devices with 512-byte pages and 64 for devices with 2048-byte pages.

spare_image

Required attribute; **read/write** access; type: [os] or object. The image object containing the spare parts of the device's pages. The image should contain the spare contents of the pages concatenated in order.

state

Optional attribute; **read/write** access; type: integer. Current state.

state_column_address

Optional attribute; **read/write** access; type: integer. Current column address.

state_counter

Optional attribute; **read/write** access; type: integer. Current sub-state.

state_page_address

Optional attribute; **read/write** access; type: integer. Current page address.

state_page_buffer

Optional attribute; **read/write** access; type: data. Current content of the page buffer.

status_byte

Optional attribute; **read/write** access; type: integer. Status Register

total_bytes_per_page

Pseudo attribute; **read/write** access; type: integer. Total number of bytes per page (Read Only).

write_protect

Optional attribute; **read/write** access; type: integer. The current state of the "write protect" pin, true (non-zero) for high (write-protect off) and false (zero) for low (write-protect on).

Command List

Commands

info print information about the device

status print status of the device

Command Descriptions

<onfi_flash>.info

Synopsis

<onfi_flash>.info

Description

Print detailed information about the configuration of the device.

<onfi_flash>.status

Synopsis

<onfi_flash>.status

Description

Print detailed information about the current status of the device.

PCF8582C

Provided By
[PCF8582C](#)

Interfaces Implemented

`conf_object`, [i2c_device](#), [i2c_slave_v2](#), [log_object](#)

Description

The PCF8582C class implements the functionality of a Philips 256 x 8 bit CMOS EEPROM with an I2C interface.

Attributes

address

Optional attribute; **read/write** access; type: `integer`. 7-bit address to be used on the I2C bus.

address_bits

Optional attribute; **read/write** access; type: `integer`. Number of address bits (8 or 16) in the eeprom.

address_mask

Optional attribute; **read/write** access; type: `integer`. Address mask to be used (together with the *address* attribute) when the device registers with the bus.

i2c_bus

Optional attribute; **read/write** access; type: `[os], object, or nil`. The I2C bus to which the device is connected.

i2c_state

Optional attribute; **read/write** access; type: `string`. Current I2C state.

image

Required attribute; **read/write** access; type: `[os] or object`. Image that will hold the memory contents.

memory_address

Optional attribute; **read/write** access; type: `integer`. Current memory address.

operation

Optional attribute; **read/write** access; type: `string`. Current operation performed.

pci-bus

Provided By
[pci-bus](#)

Interfaces Implemented

`conf_object`, [io_memory](#), [log_object](#)

Description

The pci-bus device models a logical *Peripheral Component Interconnect* bus that PCI devices can be connected to. The bus needs memory-space objects connected, representing the three PCI address spaces 'configuration', 'I/O' and 'memory'. The bus itself should be connected to a PCI bridge device.

Attributes

bridge

Required attribute; **read/write** access; type: `object`. Upstream PCI bridge object, which must implement the `pci_bridge` interface.

bus_number

Optional attribute; **read/write** access; type: `integer`. PCI bus number of this bus. In most cases this is 0, but some exceptions exist. This attribute is set by the bridge and should not be changed by the user.

conf_space

Required attribute; **read/write** access; type: `object`. Memory-space object representing the PCI configuration space.

interrupt

Optional attribute; **read/write** access; type: `[o*], object, or nil`. One or more interrupt device objects that implements the `pci_interrupt` interface. This attribute only has to be set if the bridge does not handle PCI interrupts.

io_space

Required attribute; **read/write** access; type: `object`. Memory-space object representing the PCI I/O space.

memory_space

Required attribute; **read/write** access; type: `object`. Memory-space object representing the PCI memory space.

pci_devices

Optional attribute; **read/write** access; type: `[[io]| [ioi]*]. ((id, function, object)*)`. *id* is the PCI device id. *function* is the PCI device logical function. *object* is the PCI device itself, which must implement the 'pci-device' interface. The tuple may also contain an additional integer that tells if the PCI device is enabled or not. As default the PCI device is enabled.

send_interrupt_to_bridge

Optional attribute; **read/write** access; type: **integer**. If non-zero (default), interrupt will be routed to the PCI bridge as well as the interrupt devices.

sub_bus_number

Optional attribute; **read/write** access; type: **integer**. Subordinate PCI bus number for this bus. This attribute is set by the bridge and should not be changed by the user.

upstream_target

Optional attribute; **read/write** access; type: **[os], object, or nil**. If non-NUL, all upstream transactions are forwarded to the specified object using the **pci_upstream** interface.

Command List

Commands

info print information about the device

Command Descriptions

<pci-bus>.info

Synopsis

<pci-bus>.info

Description

Print detailed information about the configuration of the device.

pcie-bus

Provided By
[pcie-bus](#)

Aliases

pcie-switch

Interfaces Implemented

`conf_object, io_memory, log_object, pci_express`

Description

The pcie-bus device models a logical *PCI Express* switch that PCI Express devices can be connected to. The switch needs memory-space objects connected, representing the three PCI Express address spaces 'configuration', 'I/O' and 'memory'. The switch itself should be connected to a PCI Express bridge device.

Attributes

bridge

Required attribute; **read/write** access; type: `object`. Upstream PCI bridge object, which must implement the `pci_bridge` and the `pci_express` interface.

bus_number

Optional attribute; **read/write** access; type: `integer`. PCI bus number of this bus. In most cases this is 0, but some exceptions exist. This attribute is set by the bridge and should not be changed by the user.

conf_space

Required attribute; **read/write** access; type: `object`. Memory-space object representing the PCI configuration space.

interrupt

Optional attribute; **read/write** access; type: `[o*], object, or nil`. One or more interrupt device objects that implements the `pci_interrupt` interface. This attribute only has to be set if the bridge does not handle PCI interrupts.

io_space

Required attribute; **read/write** access; type: `object`. Memory-space object representing the PCI I/O space.

memory_space

Required attribute; **read/write** access; type: `object`. Memory-space object representing the PCI memory space.

pci_devices

Optional attribute; **read/write** access; type: `[[iio]| [iioi]*]. ((id, function, object)*)`. *id* is the PCI device id. *function* is the PCI device logical function. *object* is the PCI device

itself, which must implement the 'pci-device' interface. The tuple may also contain an additional integer that tells if the PCI device is enabled or not. As default the PCI device is enabled.

send_interrupt_to_bridge

Optional attribute; **read/write** access; type: **integer**. If non-zero (default), interrupt will be routed to the PCI bridge as well as the interrupt devices.

sub_bus_number

Optional attribute; **read/write** access; type: **integer**. Subordinate PCI bus number for this bus. This attribute is set by the bridge and should not be changed by the user.

upstream_target

Optional attribute; **read/write** access; type: **[os], object, or nil**. If non-NUL, all upstream transactions are forwarded to the specified object using the pci_upstream interface.

Command List

Commands

info print information about the device

Command Descriptions

<pcie-bus>.info

Synopsis

<pcie-bus>.info

Description

Print detailed information about the configuration of the device.

perfanalyze-client

Provided By

[Simics Core](#)

Interfaces Implemented

[conf_object](#), [log_object](#)

Description

Performance and memory analysis class.

persistent-ram

Provided By

[Simics Core](#)

Interfaces Implemented

`conf_object`, [log_object](#)

Description

Persistent random access memory.

Attributes

image

Required attribute; **read/write** access; type: `object`. Object holding data. Must implement the image interface.

Command List

Commands

[info](#) print information about the device
[status](#) print status of the device

Command Descriptions

<persistent-ram>.info

Synopsis

`<persistent-ram>.info`

Description

Print detailed information about the configuration of the device.

<persistent-ram>.status

Synopsis

`<persistent-ram>.status`

Description

Print detailed information about the current status of the device.

PMC1553

Provided By
[PMC1553](#)

Interfaces Implemented

`conf_object, io_memory, log_object, ms1553_terminal`

Ports

HRESET ([signal](#)), SRESET ([signal](#)), bc ([int_register, io_memory](#)), pci_config ([int_register, io_memory](#)), pmc ([int_register, io_memory](#)), rt ([int_register, io_memory](#)), summit ([int_register, io_memory](#))

Description

PMC1553 is a model of the $S\mu$ MMIT-based MIL-STD-1553 module from Alphi Technology.

Attributes

AB_STD

Optional attribute; **read/write** access; type: `integer`. Select the standard, 0 for 1553B and 1 for 1553A

AUTOEN

Optional attribute; **read/write** access; type: `boolean`. Enable auto-initialization

EXT_TCLK

Optional attribute; **read/write** access; type: `integer`. Frequency in Hz of external TCLK

LOCK

Optional attribute; **read/write** access; type: `boolean`. Assert LOCK to prevent modification of status word

MSEL

Optional attribute; **read/write** access; type: `integer`. $S\mu$ MMIT Mode Selection: 0 for BC and 1 for RT.

RTA

Optional attribute; **read/write** access; type: `integer`. Remote Terminal Address

SSYSF

Optional attribute; **read/write** access; type: `boolean`. Subsystem Flag

cmd_info

Optional attribute; **read/write** access; type: `dictionary`. Current command information

config_registers

Pseudo attribute; **read-only** access; type: [i*]. The PCI configuration registers, each 32 bits in size.

expansion_rom_size

Optional attribute; **read/write** access; type: integer. The size of the expansion ROM mapping.

link

Optional attribute; **read/write** access; type: [o|os]|n{2}. The MIL-STD-1553 bus/link that the device is connected to.

pci_bus

Required attribute; **read/write** access; type: [os] or object. The PCI bus this device is connected to, implementing the pci-bus interface.

pci_config_command

Optional attribute; **read/write** access; type: integer. The PCI command register.

pci_config_device_id

Optional attribute; **read/write** access; type: integer. The Device ID of the PCI device

pci_config_vendor_id

Optional attribute; **read/write** access; type: integer. The Vendor ID of the PCI device

rom_image

Optional attribute; **read/write** access; type: [os], object, or nil. Image holding the contents for auto-initialization, the required size of bytes is

- 1088 for RT mode,
- 64 for MT mode,
- 64 + [N x 16] for BC mode, N: number of command block

sram

Required attribute; **read/write** access; type: [os] or object. Object for dual ported SRAM. The image of this object must be the object specified in the sram_image attribute.

sram_image

Required attribute; **read/write** access; type: [os] or object. Image holding SRAM contents. The size must be 128KiB. This object must be the same as the image for the object specified in the sram attribute.

Command List

Commands

hard-reset	hard reset
info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
soft-reset	soft reset
status	print status of the device

Command Descriptions

<PMC1553>.hard-reset

Synopsis

<PMC1553>.hard-reset

Description

hard reset pmc1553 device

<PMC1553>.info

Synopsis

<PMC1553>.info

Description

Print detailed information about the configuration of the device.

<PMC1553>.pci-header — *deprecated*

Synopsis

<PMC1553>.pci-header [-v]

Description

This command is deprecated; use [**<PMC1553>.print-pci-config-reg**](#)s instead.

<PMC1553>.print-pci-config-reg

Synopsis

<PMC1553>.print-pci-config-reg [-v]

Description

Print the PCI configuration registers. The *-v* flag turns on verbose mode.

<PMC1553>.soft-reset

Synopsis

<PMC1553>.soft-reset

Description

soft reset pmc1553 device

<PMC1553>.status

Synopsis

<PMC1553>.status

Description

Print detailed information about the current status of the device.

port-forward-ingcoming-server

Provided By
service-node

Interfaces Implemented
conf_object, [log_object](#)

Description

Port forwarding TCP service. Handles connections initiated external to Simics wanting access to the simulated environment.

Attributes

add_connection

Pseudo attribute; **write-only** access; type: [ssisi]. (protocol, in_addr, in_port, forward_ip, forward_port). Add a port forwarding listening port in_host on ip in_addr of the host machine and forwarding to port forward_port on ip forward_ip within the simulated environment. Protocol must be either tcp or udp.

algs

Optional attribute; **read/write** access; type: [o*]. Application Level Gateway (ALG) objects providing protocol specific support for port forwarding. The first object returning non-null in a call to its connection method will be the one that handles that particular connection.

connections

Optional attribute; **read/write** access; type: [[ssisi]*]. ((protocol, in_addr, in_port, forward_ip, forward_port)*). Sets up port forwarding listening to port in_host on ip in_addr of the host machine and forwarding to port forward_port on ip forward_ip within the simulated environment. Protocol must be either tcp or udp. BUG: If the in_addr is set to "::", the behavior is currently host-specific.

tcp

Required attribute; **read/write** access; type: object. TCP layer. Must implement the tcp interface.

temporary_connections

Pseudo attribute; **read-only** access; type: [[sisi]*]. ((protocol, in_port, forward_ip, forward_port)*). The format of this attribute is identical to the connections attribute. This attribute contains temporary port mappings that are set up by the NAPT protocol handler.

udp

Required attribute; **read/write** access; type: object. UDP layer. Must implement the udp interface.

port-forward-outgoing-server

Provided By
[service-node](#)

Interfaces Implemented
`conf_object`, [log_object](#)

Description

Port forwarding service. Handled connections originating from inside the simulated environment.

Attributes

algs

Optional attribute; **read/write** access; type: `[o*]`. Application Level Gateway (ALG) objects providing protocol specific support for port forwarding. The first object returning non-null in a call to its connection method will be the one that handles that particular connection.

connections

Optional attribute; **read/write** access; type: `[[si]]|[sissi]*`. Configures how connections are forwarded. The first two elements in each sublist specify the protocol and port to listen on. If zero, then all ports are listened on. If a sublist has five elements, then the additional arguments specify the ip address to listen on, and the ip address and port to forward to. If a sublist only has two element, then the connection will be forwarded to the ip address and port given in the incoming ip packet.

tcp

Required attribute; **read/write** access; type: `object`. TCP layer. Must implement the `tcp` interface.

udp

Required attribute; **read/write** access; type: `object`. UDP layer. Must implement the `udp` interface.

Command List

Commands

[status](#) print status of the device

Command Descriptions

`<port-forward-outgoing-server>.status`

Synopsis

`<port-forward-outgoing-server>.status`

Description

Print detailed information about the current status of the device.

port-space

Provided By
Simics Core

Interfaces Implemented

conf_object, log_object, map_demap, port_space

Description

Class providing a 16-bit+3 port address space. In response to an address being asserted on the bus, an actual EISA or ISA device signals how large a transfer it can handle through the I/O chip select 16 line (ISA), and the EX16 and EX32 lines (EISA). This allows for example a 2-byte access at address A to have different meaning than two 1-byte accesses at address A and A+1. In the port-space class, the maximal width for each address should be specified in the map attribute. Mapping 2 bytes at address A also automatically maps 1 byte at address A+1. In the same way, mapping 4 bytes at A also maps 1 byte at A+1, 2 bytes at A+2, and 1 byte at A+3.

Attributes

default_target

Optional attribute; **read/write** access; type: [oio|n] or nil. (*object, function, offset, target*). An access not targeting any object in the map attribute will be forwarded to the default target. *target* should be non-nil if and only if *object* is a translation object.

map

Optional attribute; **read/write** access; type: [[io|[os|n]iii]*]. ((*base, object, function, offset, length*)*). Maps zero or more object to the port space. *object*, which must implement the io-memory interface, is mapped from *base* to *base + length - 1* (inclusive). Both *function* and *offset* is handed on to the mapped *object* through the io-memory interface. A mapped region larger than 4 bytes will be internally split into multiple 4-byte mappings. Mappings larger than 4 bytes must be a multiple of 4 bytes long, and must start at a 4-byte aligned address.

Command List

Commands

add-map	map device in a port-space
del-map	remove device map from a port-space
get	get value of physical address without side-effects
info	print information about the device
map	list port map
read	get value of physical address
set	set physical address to specified value without side-effects
status	print status of the device
write	set physical address to specified value

Command Descriptions

<port-space>.add-map

Synopsis

```
<port-space>.add-map device base length [function] [offset]
```

Description

Map *device* into a port-space at address *base* and with length *length*. Different mappings of the same device may be identified by a device specific *function* number. The mapping may specify an offset into the device's memory space, using the *offset* argument.

See Also

[`<port-space>.map`](#), [`<port-space>.del-map`](#)

<port-space>.del-map

Synopsis

```
<port-space>.del-map device [function]
```

Description

Remove the mapping of *device* from a port-space. If a function number is given by the *function* argument, then only mappings with a matching number is removed.

See Also

[`<port-space>.map`](#), [`<port-space>.add-map`](#)

<port-space>.get

Synopsis

```
<port-space>.get address [size] [-l] [-b]
<memory-space>.get address [size] [-l] [-b]
<memory-space>.read address [size] [-l] [-b]
<port-space>.read address [size] [-l] [-b]
get address [size] [-l] [-b]
```

Description

Get value of physical memory location. The *size* argument specifies how many bytes should be read. This defaults to 4, but can be any number of bytes between 1 and 8 (inclusive) for memory-spaces and between 1 and 8 (inclusive) for port-spaces.

The *-l* and *-b* flags are used to select little-endian and big-endian byte order, respectively. If neither is given, the byte order of the currently selected processor is used.

The **get** command performs the access in inquiry mode without triggering any side-effects while **read** may trigger side-effects.

The non-namespace version of this command operates on the physical memory associated with the current processor.

See Also

[x](#), [set](#), [signed](#)

<port-space>.info

Synopsis

<port-space>.info

Description

Print detailed information about the configuration of the device.

<port-space>.map

Synopsis

<port-space>.map

Description

Prints the port map of the port space object, one line per entry in the map attribute of the port space. The *base* column is the starting address of the map. The *object* column contains the object mapped at that address. *fn* is the function number and *offs* is the offset for the object. *length* is the number of bytes mapped.

See Also

[<port-space>.add-map](#), [<port-space>.del-map](#)

<port-space>.read

Synopsis

```
<port-space>.read address [size] [-l] [-b]
<memory-space>.get address [size] [-l] [-b]
<memory-space>.read address [size] [-l] [-b]
<port-space>.get address [size] [-l] [-b]
get address [size] [-l] [-b]
```

Description

Get value of physical memory location. The size argument specifies how many bytes should be read. This defaults to 4, but can be any number of bytes between 1 and 8 (inclusive) for memory-spaces and between 1 and 8 (inclusive) for port-spaces.

The *-l* and *-b* flags are used to select little-endian and big-endian byte order, respectively. If neither is given, the byte order of the currently selected processor is used.

The **get** command performs the access in inquiry mode without triggering any side-effects while **read** may trigger side-effects.

The non-namespace version of this command operates on the physical memory associated with the current processor.

See Also

[x](#), [set](#), [signed](#)

<port-space>.set

Synopsis

```
<port-space>.set address value [size] [-l] [-b]
<memory-space>.set address value [size] [-l] [-b]
<memory-space>.write address value [size] [-l] [-b]
<port-space>.write address value [size] [-l] [-b]
set address value [size] [-l] [-b]
```

Description

Set the *size* bytes of physical memory at location *address* to *value*. The default *size* is 4 bytes, but can be anywhere between 1 and 8 (inclusive) for memory-spaces and between 1 and 4 (inclusive) for port-spaces.

If *value* is larger than the specified size, behavior is undefined.

The *-l* and *-b* flags are used to select little-endian and big-endian byte order, respectively. If neither is given, the byte order of the currently selected processor is used.

The **set** command performs the access in inquiry mode without triggering any side-effects while **write** may trigger side-effects.

If the *value* argument is a list, then each item is written to memory as a value of *size* bytes, starting at *address*. The byte order flags operate on each item in the list.

The non-namespace version of this command operates on the physical memory associated with the current processor.

See Also

[get](#), [x](#), [pselect](#)

<port-space>.status

Synopsis

```
<port-space>.status
```

Description

Print detailed information about the current status of the device.

<port-space>.write

Synopsis

```
<port-space>.write address value [size] [-l] [-b]
<memory-space>.set address value [size] [-l] [-b]
<memory-space>.write address value [size] [-l] [-b]
<port-space>.set address value [size] [-l] [-b]
set address value [size] [-l] [-b]
```

Description

Set the *size* bytes of physical memory at location *address* to *value*. The default *size* is 4 bytes, but can be anywhere between 1 and 8 (inclusive) for memory-spaces and between 1 and 4 (inclusive) for port-spaces.

If *value* is larger than the specified size, behavior is undefined.

The **-l** and **-b** flags are used to select little-endian and big-endian byte order, respectively. If neither is given, the byte order of the currently selected processor is used.

The **set** command performs the access in inquiry mode without triggering any side-effects while **write** may trigger side-effects.

If the *value* argument is a list, then each item is written to memory as a value of *size* bytes, starting at *address*. The byte order flags operate on each item in the list.

The non-namespace version of this command operates on the physical memory associated with the current processor.

See Also

[get](#), [x](#), [pselect](#)

preferences

Provided By

[Simics Core](#)

Interfaces Implemented

`conf_object`, [log_object](#)

Description

The `preferences` class represent Simics's global simulator preferences that, unlike the settings in the `sim` object, usually are the same between sessions.

Attributes

async_command_line

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE (default) if command lines in Simics should be asynchronous, meaning that a prompt exists and commands can be entered while the simulation is advancing.

autostart_sim

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if the simulation should automatically start when a new session is started. Default FALSE.

command_log

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if command should be logged to the command log file. The value for the current session is in `sim->command_log`.

compress_images

Optional attribute; **read/write** access; type: `boolean`. Determines if images saved by write-configuration or save-persistent-state are compressed or not by default. Using the -z or -u flag will override this preference.

console_in_gui

Optional attribute; **read/write** access; type: `boolean`. When set to TRUE, graphics consoles will be integrated with the GUI. Default FALSE.

cpu_mode

Optional attribute; **read/write** access; type: `string`. Default CPU mode used when loading processor modules. One of normal and stall. The value for the current session is in `sim->cpu_mode`.

enable_multithreading

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if multithreading should be enabled when Simics is started. Default FALSE.

enable_rev_exec

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if reverse execution should be enabled when Simics is started. Default FALSE.

gui_cmd_line_font

Optional attribute; **read/write** access; type: string or nil. Font used in the command line window.

gui_default_workspace

Optional attribute; **read/write** access; type: string or nil. Default Simics workspace to open when Simics is launched in GUI mode. The value for the current session is in sim->workspace.

gui_mode

Optional attribute; **read/write** access; type: string. Default GUI mode when starting the command line version of Simics. The value is one of “mixed” and “no-gui”. The value for the current session is in sim->gui_mode.

gui_open cmdline_win

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the command line window always should open when Simics is launched in GUI mode.

gui_open control win

Optional attribute; **read/write** access; type: boolean. Set to TRUE if the control window should open when Simics is launched in command line mode and windows are allowed.

gui_source_code_encoding

Optional attribute; **read/write** access; type: string or nil. Default encoding used in the source code window.

history_lines

Optional attribute; **read/write** access; type: integer. The number of input lines saved in the command history.

legacy_cell

Optional attribute; **read/write** access; type: boolean. Allow pre-cell configurations to run.

legacy_conf

Optional attribute; **read/write** access; type: boolean. (This attribute is deprecated.) All object IDs are accessible in the global Python conf namespace and all objects will be accessible in the global namespace for many CLI commands when this attribute is TRUE. When set to FALSE, objects are only accessible from the component hierarchy. Default is FALSE.

legacy_global_id

Optional attribute; **read/write** access; type: boolean. (This attribute is deprecated.) When set to true, the object name of a link is used as a global identifier in distributed simulations to figure out what links are distributed unless the link has the global_id attribute set. When set to false (default), only the global_id attribute is used as a global identifier.

legacy_name

Optional attribute; **read/write** access; type: `boolean`. (This attribute is deprecated.) Objects will have the name given when created if this attribute is TRUE. The object names are identical to their hierarchical location if this attribute is FALSE. Default value is FALSE. This attribute should be set to TRUE when using old scripts that expects non-hierarchical object names.

license_file

Optional attribute; **read/write** access; type: `string` or `nil`. The path to the default FlexNET license file. See also the `license_file` attribute in the `sim` class. The value for the current session is in `sim->license_file`. Simics has to be restarted for a new license path to take effect.

network_iface_helper

Optional attribute; **read/write** access; type: `string` or `nil`. Full path name of helper executable that opens the host network interface. The same helper is used for both the real-network-router and real-network-bridge classes. The helper executable must have permissions to open the network interface.

output_grouping

Optional attribute; **read/write** access; type: `[iiii]`. The default digit grouping for integers in the command line, for binary, octal, decimal and hexadecimal integers.

output_radix

Optional attribute; **read/write** access; type: `integer`. The default output radix for integers in the command line. One of 2, 8, 10 and 16. Default is 10.

progress_cnt

Optional attribute; **read/write** access; type: `string`. The type of progress indication in the GUI. Valid values are “instructions”, “virtual-time”, and “virtual-load”.

readline_shortcuts

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if GNU Readline style keyboard shortcuts should be used in the command line. If set to FALSE, Windows style shortcuts are used.

swap_dir

Optional attribute; **read/write** access; type: `string`. Directory where to store swap files, to limit Simics memory usage.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<preferences>.info

Synopsis

<preferences>.info

Description

Print detailed information about the configuration of the device.

<preferences>.status

Synopsis

<preferences>.status

Description

Print detailed information about the current status of the device.

ram

Provided By

[Simics Core](#)

Interfaces Implemented

[conf_object](#), [log_object](#)

Description

Random access memory.

Attributes

image

Required attribute; **read/write** access; type: `object`. Object holding data. Must implement the image interface.

Command List

Commands

[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

<ram>.info

Synopsis

`<ram>.info`

Description

Print detailed information about the configuration of the device.

<ram>.status

Synopsis

`<ram>.status`

Description

Print detailed information about the current status of the device.

rapidio_link_endpoint

Provided By

[rapidio-link](#)

Interfaces Implemented

`conf_object`, [log_object](#), [rapidio_v5](#)

Ports

`inject` ([ethernet_common](#))

Description

RapidIO link endpoint

Attributes

device

Required attribute; **read/write** access; type: `[os]`, `object`, or `nil`. The device connected to this endpoint.

id

Required attribute; **read/write** access; type: `integer`. Endpoint ID. The ID of each endpoint must be unique among the link's endpoints, and it may not be 0 or `0xffffffffffffffffffff`

link

Required attribute; **read/write** access; type: `object`. The link object to which this endpoint belongs.

Command List

Commands

[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

`<rapidio_link_endpoint>.info`

Synopsis

`<rapidio_link_endpoint>.info`

Description

Print detailed information about the configuration of the device.

`<rapidio_link_endpoint>.status`

Synopsis

`<rapidio_link_endpoint>.status`

Description

Print detailed information about the current status of the device.

rapidio_link_impl

Provided By

[rapidio-link](#)

Interfaces Implemented

conf_object, [log_object](#)

Description

Point-to-point link for rapidio_v5

Attributes

goal_latency

Required attribute; **read/write** access; type: float. The desired latency for this link.

Command List

Commands

info print information about the device

status print status of the device

Command Descriptions

<rapidio_link_impl>.info

Synopsis

<rapidio_link_impl>.info

Description

Print detailed information about the configuration of the device.

<rapidio_link_impl>.status

Synopsis

<rapidio_link_impl>.status

Description

Print detailed information about the current status of the device.

rapidio_simple_device

Provided By

[rapidio-simple-device](#)

Interfaces Implemented

`conf_object`, [io_memory](#), [log_object](#), [rapidio_v3](#)

Ports

`regs` ([int_register](#), [io_memory](#)), `trigger_dma` ([signal](#))

Description

Dummy RapidIO device.

Attributes

peer

Optional attribute; **read/write** access; type: `[os]`, `object`, or `nil`. The connected RapidIO switch or device

rapidio_tape

Provided By

[rapidio-tape](#)

Interfaces Implemented

[conf_object](#), [log_object](#), [rapidio_v3](#)

Description

RapidIO device that just records the incoming transactions, or plays back transactions from a file.

Command List

Commands

info	print information about the device
playback-stop	stop traffic generation
recorder-stop	stop traffic recording
status	print status of the device

Command Descriptions

<rapidio_tape>.info

Synopsis

[<rapidio_tape>.info](#)

Description

Print detailed information about the configuration of the device.

<rapidio_tape>.playback-stop

Synopsis

[<rapidio_tape>.playback-stop](#)

Description

Stop bus traffic generation previously started with [new-rapidio-tape](#).

See Also

[<rapidio_tape>.recorder-stop](#)

<rapidio_tape>.recorder-stop

Synopsis

[<rapidio_tape>.recorder-stop](#)

Description

Stop bus traffic recording previously started with [new-rapidio-tape](#).

See Also

[`<rapidio_tape>.playback-stop`](#)

`<rapidio_tape>.status`

Synopsis

`<rapidio_tape>.status`

Description

Print detailed information about the current status of the device.

realtime

Provided By
[realtime](#)

Interfaces Implemented
conf_object, [log_object](#)

Description

In some cases simulated time may run faster than real time; this can happen if the OS is in a tight idle loop or an instruction halts execution waiting for an interrupt, or if the host machine is simply sufficiently fast. This can cause problems for programs that interact with the real world (for example the user), since time-outs may expire really fast. A **realtime** object will, when enabled, periodically check the simulation speed and wait for a while if it is too high.

Attributes

check_interval

Session attribute; **read/write** access; type: `integer`. How frequently elapsed simulated time should be compared to elapsed real time. Specified in milliseconds of simulated time; the default is 100. The actual time between comparisons may be less than this. It will never be less than **cpu-switch-time**, so check that setting if you need very fine-grained realtime behavior.

clock_object

Required attribute; **read/write** access; type: `object`. The object used to measure simulated time. This can be any processor in the system.

drift_compensate

Session attribute; **read/write** access; type: `float`. The *speed* attribute says how fast the simulation *should* run, but the actual speed will always deviate a little from that value even if the host is fast enough. To keep these errors from accumulating, the simulation speed has to be adjusted; *drift-compensate* regulates how much it may be adjusted. If set to (for example) 0.25, simulation speed may be increased or decreased by up to 25% if necessary to make up for any accumulated drift with respect to real time. If set to zero (the default), the simulation speed may not be changed at all from its set value.

enabled

Session attribute; **read/write** access; type: `boolean`. Whether the real-time behavior is enabled or not. Defaults to false.

max_oversleep

Session attribute; **read/write** access; type: `integer`. The sleep system call will usually sleep somewhat longer than requested. To compensate for this, Simics will ask for a shorter sleep, and busy-wait the rest of the time. This parameter determines how much shorter, in microseconds. (This will only help against delays imposed by the sleep implementation, not against delays caused by other processes on the system.)

rtc_freq

Pseudo attribute; **read-only** access; type: `integer`. Frequency of the realtime clock, in Hz.

speed

Session attribute; **read/write** access; type: `float`. How fast the simulated time runs compared to real time. The default is 1, which means that simulated time runs at the same speed as real time. Note that Simics may be unable to run as fast as requested if the host is not fast enough.

Command List

Commands

<code>disable</code>	disable real-time behavior
<code>enable</code>	enable real-time behavior
<code>info</code>	print information about the device
<code>status</code>	print status of the device

Command Descriptions

`<realtime>.disable`

Synopsis

```
<realtime>.disable
<realtime>.enable [speed] [check-interval] [drift-compensate]
disable-real-time-mode
enable-real-time-mode [speed] [check_interval] [drift-compensate]
```

Description

In some cases simulated time may run faster than real time; this can happen if the OS is in a tight idle loop or an instruction halts execution waiting for an interrupt, or if the host machine is simply sufficiently fast. This can cause problems for programs that interact with the real world (for example the user), since time-outs may expire really fast.

A **realtime** object will, when enabled, periodically check the simulation speed and wait for a while if it is too high. *speed* specifies how fast simulated time is allowed to run, in percent of real time; default is 100. *check-interval* specifies how often the check should take place, in milliseconds of simulated time; default is 1000. Higher values give better performance; lower values reduce the maximum difference between real and simulated time. Setting this to less than **cpu-switch-time** has no effect.

The *speed* argument says how fast the simulation *should* run, but the actual speed will always deviate a little from that value even if the host is fast enough. To keep these errors from accumulating, the simulation speed has to be adjusted; *drift-compensate* regulates how much it may be adjusted. If set to (for example) 0.25, simulation speed may be increased or decreased by up to 25% if necessary to make up for any accumulated drift with respect to real time. If set to zero (the default), the simulation speed may not be changed at all from its set value.

enable and **disable** work on a given **realtime** object; **enable-real-time-mode** and **disable-real-time-mode** work on a default **realtime** object.

<realtime>.enable

Synopsis

```
<realtime>.enable [speed] [check-interval] [drift-compensate]
<realtime>.disable
disable-real-time-mode
enable-real-time-mode [speed] [check_interval] [drift_compensate]
```

Description

In some cases simulated time may run faster than real time; this can happen if the OS is in a tight idle loop or an instruction halts execution waiting for an interrupt, or if the host machine is simply sufficiently fast. This can cause problems for programs that interact with the real world (for example the user), since time-outs may expire really fast.

A **realtime** object will, when enabled, periodically check the simulation speed and wait for a while if it is too high. *speed* specifies how fast simulated time is allowed to run, in percent of real time; default is 100. *check-interval* specifies how often the check should take place, in milliseconds of simulated time; default is 1000. Higher values give better performance; lower values reduce the maximum difference between real and simulated time. Setting this to less than **cpu-switch-time** has no effect.

The *speed* argument says how fast the simulation *should* run, but the actual speed will always deviate a little from that value even if the host is fast enough. To keep these errors from accumulating, the simulation speed has to be adjusted; *drift-compensate* regulates how much it may be adjusted. If set to (for example) 0.25, simulation speed may be increased or decreased by up to 25% if necessary to make up for any accumulated drift with respect to real time. If set to zero (the default), the simulation speed may not be changed at all from its set value.

enable and **disable** work on a given **realtime** object; **enable-real-time-mode** and **disable-real-time-mode** work on a default **realtime** object.

<realtime>.info

Synopsis

```
<realtime>.info
```

Description

Print detailed information about the configuration of the device.

<realtime>.status

Synopsis

```
<realtime>.status
```

Description

Print detailed information about the current status of the device.

recorder

Provided By
[recorder](#)

Interfaces Implemented

`conf_object`, `log_object`, `recorder`, `recorder_v2`

Description

The `recorder` class is used to record and playback I/O events. Recordable events include network traffic and mouse and keyboard events.

Command List

Commands

<code>info</code>	print information about the device
<code>playback-start</code>	play back recorded I/O
<code>playback-stop</code>	stop playback
<code>recorder-save</code>	save recorder
<code>recorder-start</code>	record input to file
<code>recorder-stop</code>	stop recorder
<code>status</code>	print status of the device

Command Descriptions

`<recorder>.info`

Synopsis

`<recorder>.info`

Description

Print detailed information about the configuration of the device.

`<recorder>.playback-start`

Synopsis

`<recorder>.playback-start` *input-file*

Description

Starts playback from specified file. The I/O traffic in the file that is to be played back must have been recorded using a machine configuration identical to the current configuration. It is highly recommended that console input is blocked when you play back I/O, or the session may lose its synchronization.

`<recorder>.playback-stop`

Synopsis

```
<recorder>.playback-stop
```

Description

Stop the I/O playback. Note that once playback has been stopped it cannot be restarted.

<recorder>.recorder-save**Synopsis**

```
<recorder>.recorder-save output-file
```

Description

Save recorded playback data.

<recorder>.recorder-start**Synopsis**

```
<recorder>.recorder-start output-file
```

Description

Record input data to specified file. I/O is recorded from all devices that are recording aware (that includes all devices that are shipped in standard distributions).

<recorder>.recorder-stop**Synopsis**

```
<recorder>.recorder-stop
```

Description

Stop the recording of incoming data.

<recorder>.status**Synopsis**

```
<recorder>.status
```

Description

Print detailed information about the current status of the device.

remote_sync_domain

Provided By

[Simics Core](#)

Interfaces Implemented

`conf_object`, [log_object](#)

Description

A controller object for a group of synchronized nodes.

Attributes

server

Required attribute; **read/write** access; type: `string`. The address of the server.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<remote_sync_domain>.info

Synopsis

`<remote_sync_domain>.info`

Description

Print detailed information about the configuration of the device.

<remote_sync_domain>.status

Synopsis

`<remote_sync_domain>.status`

Description

Print detailed information about the current status of the device.

remote_sync_node

Provided By

[Simics Core](#)

Interfaces Implemented

[conf_object](#), [log_object](#)

Description

A controller object for a group of synchronized nodes.

Command List

Commands

[info](#) print information about the device

[status](#) print status of the device

Command Descriptions

<remote_sync_node>.info

Synopsis

[<remote_sync_node>.info](#)

Description

Print detailed information about the configuration of the device.

<remote_sync_node>.status

Synopsis

[<remote_sync_node>.status](#)

Description

Print detailed information about the current status of the device.

remote_sync_server

Provided By

[Simics Core](#)

Interfaces Implemented

`conf_object`, [log_object](#)

Description

A server for connecting to a synchronization domain.

Attributes

domain

Required attribute; **read/write** access; type: `object`. Synchronization domain clients are connected to.

port

Required attribute; **read/write** access; type: `integer`. Port listened to. If 0, an available port is chosen.

Command List

Commands

[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

<remote_sync_server>.info

Synopsis

`<remote_sync_server>.info`

Description

Print detailed information about the configuration of the device.

<remote_sync_server>.status

Synopsis

`<remote_sync_server>.status`

Description

Print detailed information about the current status of the device.

rev-execution

Provided By

[Simics Core](#)

Interfaces Implemented

`conf_object`, `image_snoop`, [log_object](#)

Description

Provides support for reverse execution.

rn-eth-bridge-raw

Provided By
[real-network](#)

Interfaces Implemented

`conf_object`, [ethernet_common](#), [ethernet_device](#), [log_object](#)

Description

Ethernet bridge using raw interface access.

Attributes

recorder

Required attribute; **read/write** access; type: `object`. Recorder device for recording and playback of traffic from the real network.

Command List

Commands

[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

`<rn-eth-bridge-raw>.info`

Synopsis

`<rn-eth-bridge-raw>.info`

Description

Print detailed information about the configuration of the device.

`<rn-eth-bridge-raw>.status`

Synopsis

`<rn-eth-bridge-raw>.status`

Description

Print detailed information about the current status of the device.

rn-eth-bridge-tap

Provided By
[real-network](#)

Interfaces Implemented

[conf_object](#), [ethernet_common](#), [ethernet_device](#), [log_object](#)

Description

Ethernet bridge using a TAP interface.

Attributes

recorder

Required attribute; **read/write** access; type: `object`. Recorder device for recording and playback of traffic from the real network.

Command List

Commands

finish-connection	Finalize a real network connection
info	print information about the device
status	print status of the device

Command Descriptions

<rn-eth-bridge-tap>.finish-connection

Synopsis

<rn-eth-bridge-tap>.finish-connection

Description

Finish the connection to the real network once the TAP interface has been assigned an IP address and network mask that the real-network object has to read. This command usually isn't needed since Simics will retry getting the interface configuration when it receives a packet from or to the real network.

<rn-eth-bridge-tap>.info

Synopsis

<rn-eth-bridge-tap>.info

Description

Print detailed information about the configuration of the device.

<rn-eth-bridge-tap>.status

Synopsis

<rn-eth-bridge-tap>.status

Description

Print detailed information about the current status of the device.

rn-eth-proxy-raw

Provided By
[real-network](#)

Interfaces Implemented

`conf_object`, [ethernet_common](#), [ethernet_device](#), [log_object](#)

Description

Ethernet proxy using raw interface access.

Attributes

recorder

Required attribute; **read/write** access; type: `object`. Recorder device for recording and playback of traffic from the real network.

Command List

Commands

[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

`<rn-eth-proxy-raw>.info`

Synopsis

`<rn-eth-proxy-raw>.info`

Description

Print detailed information about the configuration of the device.

`<rn-eth-proxy-raw>.status`

Synopsis

`<rn-eth-proxy-raw>.status`

Description

Print detailed information about the current status of the device.

rn-ip-router-raw

Provided By
[real-network](#)

Interfaces Implemented

[conf_object](#), [ethernet_common](#), [ethernet_device](#), [log_object](#)

Description

Real-network IP router using raw interface access.

Attributes

ip

Required attribute; **read/write** access; type: `string`. The IP address of the router connected to the simulated link.

recorder

Required attribute; **read/write** access; type: `object`. Recorder device for recording and playback of traffic from the real network.

Command List

Commands

[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

<rn-ip-router-raw>.info

Synopsis

<rn-ip-router-raw>.info

Description

Print detailed information about the configuration of the device.

<rn-ip-router-raw>.status

Synopsis

<rn-ip-router-raw>.status

Description

Print detailed information about the current status of the device.

rom

Provided By

[Simics Core](#)

Interfaces Implemented

`conf_object`, [log_object](#)

Description

Read-only memory.

Attributes

image

Required attribute; **read/write** access; type: `object`. Object holding data. Must implement the image interface.

Command List

Commands

`info` print information about the device
`status` print status of the device

Command Descriptions

<rom>.info

Synopsis

`<rom>.info`

Description

Print detailed information about the configuration of the device.

<rom>.status

Synopsis

`<rom>.status`

Description

Print detailed information about the current status of the device.

scsi-bus

Provided By
[scsi-bus](#)

Interfaces Implemented
conf_object, [log_object](#)

Description

The scsi-bus extension handles the interfacing between scsi devices (i.e. disks) and scsi controllers. Once a scsi-bus has been created, devices can be added to it.

Attributes

targets

Required attribute; **read/write** access; type: `[[ioi]*].((target, device, arbiting)+)` where *target* is a SCSI ID, *device* is a device object, and *arbiting* is a flag set to non-zero when the device is participating in bus arbitration.

Command List

Commands

[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

`<scsi-bus>.info`

Synopsis

`<scsi-bus>.info`

Description

Print detailed information about the configuration of the device.

`<scsi-bus>.status`

Synopsis

`<scsi-bus>.status`

Description

Print detailed information about the current status of the device.

scsi-cdrom

Provided By

[scsi-cdrom](#)

Interfaces Implemented

`conf_object`, [log_object](#)

Description

Generic CD-ROM with SCSI interface.

Attributes

max_data_width

Optional attribute; **read/write** access; type: `integer`. The maximum supported data transfer width in bytes.

may_disc

Optional attribute; **read/write** access; type: `integer`. Set to 1 if the CD-ROM may disconnect from the current session.

memory

Optional attribute; **read/write** access; type: `object`. Backdoor access: Name of a memory space object, that implements the 'lookup' interface.

ns_per_transfer

Optional attribute; **read/write** access; type: `integer`. The constant delay, in nanoseconds, for a CD-ROM transfer. This delay will be added to the size-dependent delay set by the 'ns_per_unit' attribute. Note that this is not the same as average seek time for random accesses. (This attribute will be replaced by a more detailed timing model in the future.)

ns_per_unit

Optional attribute; **read/write** access; type: `integer`. The number of nanoseconds it takes to transfer data of the size set in the 'timing_unit_size' attribute. This delay will be added to the constant delay set in the 'ns_per_transfer' attribute. (This attribute will be replaced by a more detailed timing model in the future.)

reselection_support

Optional attribute; **read/write** access; type: `integer`. Set to 1 if the CD-ROM supports reselection (not functional).

scsi_bus

Required attribute; **read/write** access; type: `object`. Name of the SCSI bus the CD-ROM is connected to. This object must implement the 'scsi-bus' interface.

scsi_target

Required attribute; **read/write** access; type: `integer`. Target ID on the SCSI bus for the CD-ROM.

selection_time

Optional attribute; **read/write** access; type: **float**. The selection time on the SCSI bus in seconds.

synchronous_support

Optional attribute; **read/write** access; type: **integer**. Set to 1 if the CD-ROM supports synchronous transfers (no-op).

timing_unit_size

Optional attribute; **read/write** access; type: **integer**. The amount of data (in bytes) that is used to calculate CD-ROM delays together with the 'ns_per_unit' attribute. (This attribute will be replaced by a more detailed timing model in the future.)

Command List

Commands

- eject** eject media from CD-ROM drive
- info** print information about the device
- insert** insert medium in CD-ROM drive
- status** print status of the device

Command Descriptions

<scsi-cdrom>.eject

Synopsis

<scsi-cdrom>.eject

Description

Eject a media from the CD-ROM drive. The media must have been previously inserted with the 'insert' command.

<scsi-cdrom>.info

Synopsis

<scsi-cdrom>.info

Description

Print detailed information about the configuration of the device.

<scsi-cdrom>.insert

Synopsis

<scsi-cdrom>.insert *medium*

Description

Insert a medium in the CD-ROM drive. The medium is the name of a CD-ROM media object, e.g. a file-cdrom object.

```
<scsi-cdrom>.status
```

Synopsis

```
<scsi-cdrom>.status
```

Description

Print detailed information about the current status of the device.

scsi-disk

Provided By
[scsi-disk](#)

Interfaces Implemented
conf_object, [log_object](#)

Description

Generic disk with SCSI interface.

Attributes

geometry

Required attribute; **read/write** access; type: `[iii]`. The geometry of the disk. ()

image

Required attribute; **read/write** access; type: `object`. Name of the image object holding the actual data of the disk. This object must implement the 'image' interface.

max_data_width

Optional attribute; **read/write** access; type: `integer`. The maximum supported data transfer width in bytes.

may_disc

Optional attribute; **read/write** access; type: `integer`. Set to 1 if the disk may disconnect from the current session.

memory

Optional attribute; **read/write** access; type: `object`. Backdoor access: Name of a memory space object, that implements the 'lookup' interface.

ns_per_transfer

Optional attribute; **read/write** access; type: `integer`. The constant delay, in nanoseconds, for a disk transfer. This delay will be added to the size-dependent delay set by the 'ns_per_unit' attribute. Note that this is not the same as average seek time for random accesses. (This attribute will be replaced by a more detailed timing model in the future.)

ns_per_unit

Optional attribute; **read/write** access; type: `integer`. The number of nanoseconds it takes to transfer data of the size set in the 'timing_unit_size' attribute. This delay will be added to the constant delay set in the 'ns_per_transfer' attribute. (This attribute will be replaced by a more detailed timing model in the future.)

reselection_support

Optional attribute; **read/write** access; type: `integer`. Set to 1 if the disk supports reselection (not functional).

scsi_bus

Required attribute; **read/write** access; type: **object**. Name of the SCSI bus the disk is connected to. This object must implement the 'scsi-bus' interface.

scsi_target

Required attribute; **read/write** access; type: **integer**. Target ID on the SCSI bus for the disk.

selection_time

Optional attribute; **read/write** access; type: **float**. The selection time on the SCSI bus in seconds.

synchronous_support

Optional attribute; **read/write** access; type: **integer**. Set to 1 if the disk supports synchronous transfers (no-op).

timing_unit_size

Optional attribute; **read/write** access; type: **integer**. The amount of data (in bytes) that is used to calculate disk delays together with the 'ns_per_unit' attribute. (This attribute will be replaced by a more detailed timing model in the future.)

Command List

Commands

add-diff-file	add a diff file to the image
add-diff-partial-file	add a partial diff file to the image
add-sun-partition	add partition from a file
create-partition	add a partition to disk
create-sun-vtoc-header	write a new VTOC to a Sun disk
create-sun-vtoc-partition	write partition data in the VTOC on a Sun disk
delete-sun-vtoc-partition	delete partition data from the VTOC on a Sun disk
dump-sun-partition	write partition as a file
info	print information about the device
print-partition-info	print info about a pc-style partition
print-partition-table	print the disk's pc-style partition table
print-sun-vtoc	print the VTOC for a Sun disk
save-diff-file	save diff file to disk
status	print status of the device

Command Descriptions

<scsi-disk>.add-diff-file

Synopsis

<scsi-disk>.add-diff-file *filename* [-replace] [-rw]

Description

Add a diff file to the list of files for an disk. The diff file was typically created with the 'save-diff-file' command, or by a saved configuration. This is basically the same command as `<image>.add-diff-file`. The file can be made writable instead of read-only using the `-rw` flag. To replace any existing files, use `-replace`.

See Also

[`<scsi-disk>.save-diff-file`](#), [`<image>.add-diff-file`](#)

`<scsi-disk>.add-diff-partial-file`

Synopsis

`<scsi-disk>.add-diff-partial-file filename start [size]`

Description

Add a diff file to the list of files for an disk. The diff file was typically created with the 'save-diff-file' command, by one of the dump-* partition commands, or by a saved configuration. This is basically the same command as `<image>.add-partial-diff-file`.

See Also

[`<scsi-disk>.save-diff-file`](#), [`<image>.add-diff-file`](#), [`<image>.add-partial-diff-file`](#)

`<scsi-disk>.add-sun-partition`

Synopsis

`<scsi-disk>.add-sun-partition number file`

Description

Adds an image or diff as a sun partition to the current disk.

See Also

[`<scsi-disk>.dump-sun-partition`](#)

`<scsi-disk>.create-partition`

Synopsis

`<scsi-disk>.create-partition partition start_cylinder start_head start_sector end_cylinder end_head end_sector (type_id|"type_name") start_lba size_in_sectors [-boot-indication]`

Description

Add a partition to a disk's partition table. Will create a primary PC-style partition with number *partition*. Any existing partition will be overwritten. The start location of the partition is given both in CHS format with *start_cylinder*, *start_head*, *start_sector*, and in linear format with *start_lba*. The partition endpoint is specified in CHS format with *end_cylinder*, *end_head*, and *end_sector*. The size in 512-byte sectors is *size_in_sectors*.

The slight overspecification of the parameters is there because they depend on how the BIOS translates the disk geometry to the BIOS geometry. All CHS coordinates given to this command should be in the translated BIOS geometry.

The partition boot bit is enabled with the *-boot-indication* flag (default off).

The partition type can either raw numerical code, or a symbolic name. The following symbolic names are understood by the command: "unused" (0x0), "DOS 12-bit FAT" (0x1), "DOS 3.0+ 16-bit FAT" (0x4), "DOS 3.3+ extended" (0x5), "DOS 3.31+ 16-bit FAT" (0x6), "OS/2 IFS" (0x7), "OS/2 boot manager" (0xa), "Win95 32-bit FAT" (0xb), "Win95 32-bit FAT, LBA" (0xc), "Win95 16-bit FAT, LBA" (0xe), "Win95 extended, LBA" (0xf), "Linux swap" (0x82), "Linux native" (0x83), "Linux extended" (0x85), "Linux LVM" (0x8e), "BSD/386" (0xa5), "OpenBSD" (0xa6), "NetBSD" (0xa9), and "BeOS" (0xeb).

See Also

[`<scsi-disk>.print-partition-table`](#)

`<scsi-disk>.create-sun-vtoc-header`

Synopsis

`<scsi-disk>.create-sun-vtoc-header [C] [H] [S] [-quiet]`

Description

Create and write a new VTOC to a Sun disk. The geometry information written is taken from the configuration attribute 'geometry' of the disk, unless specified with the **C**, **H** and **S** parameters. A new empty partition table is also created, with only the standard 'backup' partition as number 2. *-quiet* makes the command silent in case of success.

See Also

[`<scsi-disk>.print-sun-vtoc`](#), [`<scsi-disk>.create-sun-vtoc-partition`](#), [`<scsi-disk>.delete-sun-vtoc-partition`](#)

`<scsi-disk>.create-sun-vtoc-partition`

Synopsis

`<scsi-disk>.create-sun-vtoc-partition number "tag" "flag" start-block num-blocks [-quiet]`

Description

Write partition information to the VTOC on a Sun disk. This command does not change the format of the disk, and it does not create any file system on the partition. Only the 'Volume Table Of Contents' is modified. No checking is performed to make sure that partitions do not overlap, or that they do not exceed the disk size. *-quiet* makes the command silent in case of success.

See Also

[`<scsi-disk>.print-sun-vtoc`](#), [`<scsi-disk>.create-sun-vtoc-header`](#), [`<scsi-disk>.delete-sun-vtoc-partition`](#)

<scsi-disk>.delete-sun-vtoc-partition

Synopsis

<scsi-disk>.delete-sun-vtoc-partition *number* [-quiet]

Description

Delete the information in the VTOC on a Sun disk for the specified partition. No other modification on the disk is performed. -quiet makes the command silent in case of success.

See Also

[`<scsi-disk>.print-sun-vtoc`](#), [`<scsi-disk>.create-sun-vtoc-header`](#), [`<scsi-disk>.create-sun-vtoc-partition`](#)

<scsi-disk>.dump-sun-partition

Synopsis

<scsi-disk>.dump-sun-partition *number file*

Description

Write all data from a Sun disk partition to the specified file in raw format.

See Also

[`<scsi-disk>.print-sun-vtoc`](#), [`<scsi-disk>.add-sun-partition`](#)

<scsi-disk>.info

Synopsis

<scsi-disk>.info

Description

Print detailed information about the configuration of the device.

<scsi-disk>.print-partition-info

Synopsis

<scsi-disk>.print-partition-info *num [filename]*

Description

Print information about a partition. If a filename is given, then suitable partition parameters for use with image aware e2tools will be written to that file.

<scsi-disk>.print-partition-table

Synopsis

<scsi-disk>.print-partition-table

Description

Print the PC-style partition table for a disk.

See Also

[`<scsi-disk>.create-partition`](#)

`<scsi-disk>.print-sun-vtoc`**Synopsis**

`<scsi-disk>.print-sun-vtoc`

Description

Print the contents of the VTOC (volume table of contents) for a Sun disk. This is similar to the Solaris 'prtvtoc' command.

See Also

[`<scsi-disk>.create-sun-vtoc-header`](#), [`<scsi-disk>.create-sun-vtoc-partition`](#), [`<scsi-disk>.delete-sun-vtoc-partition`](#)

`<scsi-disk>.save-diff-file`**Synopsis**

`<scsi-disk>.save-diff-file filename`

Description

Writes changes to the image as a diff file in craff format. This is basically the same command as `<image>.safe-diff-file`.

See Also

[`<scsi-disk>.add-diff-file`](#), [`<image>.save-diff-file`](#)

`<scsi-disk>.status`**Synopsis**

`<scsi-disk>.status`

Description

Print detailed information about the current status of the device.

selfprof

Provided By
[selfprof](#)

Interfaces Implemented
conf_object, [log_object](#)

Description
Class used for self-profiling of Simics

Command List

Commands

- [**info**](#) print information about the device
- [**status**](#) print status of the device

Command Descriptions

<selfprof>.info

Synopsis
`<selfprof>.info`

Description
Print detailed information about the configuration of the device.

<selfprof>.status

Synopsis
`<selfprof>.status`

Description
Print detailed information about the current status of the device.

ser-link-endpoint

Provided By
[ser-link](#)

Interfaces Implemented

[conf_object](#), [log_object](#), [serial_device](#), [serial_link](#)

Description

Serial link endpoint

Attributes

device

Required attribute; **read/write** access; type: `[os], object, or nil`. The device connected to this endpoint.

id

Required attribute; **read/write** access; type: `integer`. Endpoint ID. The ID of each endpoint must be unique among the link's endpoints, and it may not be 0 or `0xffffffffffff`

link

Required attribute; **read/write** access; type: `object`. The link object to which this endpoint belongs.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<ser-link-endpoint>.info

Synopsis

<ser-link-endpoint>.info

Description

Print detailed information about the configuration of the device.

<ser-link-endpoint>.status

Synopsis

<ser-link-endpoint>.status

Description

Print detailed information about the current status of the device.

ser-link-impl

Provided By
[ser-link](#)

Interfaces Implemented
conf_object, [log_object](#)

Description
Serial link

Attributes

goal_latency
Required attribute; **read/write** access; type: float. The desired latency for this link.

Command List

Commands
[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

<ser-link-impl>.info

Synopsis
<ser-link-impl>.info

Description
Print detailed information about the configuration of the device.

<ser-link-impl>.status

Synopsis
<ser-link-impl>.status

Description
Print detailed information about the current status of the device.

serial-link

Provided By
[serial-link](#)

Interfaces Implemented
conf_object, [log_object](#), [serial_link](#)

Description
obsolete serial link

Attributes

central

Optional attribute; **read/write** access; type: object or nil. The Simics Central client object used for distributing the simulation. (or a cell)

devices

Optional attribute; **read/write** access; type: $[[i|n, i, s, o|n, [f|o|n[ii]|n]ii|n] *]$. The connected devices. It is a list of entries, where each entry is a list [*global id, local id, name, obj, info*]. The *obj* is either a reference to the device object for local devices, or nil if the device was in another Simics process. The *info* field is a list [*last-time, ready-posted, buffer*].

frequency

Optional attribute; **read/write** access; type: integer. The frequency of the clock used by the link to time stamp traffic and state changes. This defines the granularity of the time of any event on the link. The frequency is given as number of ticks per second.

latency

Optional attribute; **read/write** access; type: float or integer. The latency on the link. Every packet that is sent over this link is delayed by at least the time specified in this attribute. Other simulated state changes, such as auto-negotiation and addresses also propagate over the link with the same delay. The type of this attribute is either an integer number of clock ticks in the time base specified in the *frequency* attribute, or a floating point number which defines the latency in seconds.

link_id

Pseudo attribute; **read/write** access; type: integer. The ID number of the link this link object belongs to.

link_obj_id

Pseudo attribute; **read/write** access; type: integer. The local ID number of this link object on the link.

linkname

Optional attribute; **read/write** access; type: string or nil. The global name of the link that this link objects belongs to. This name is used to identify links that are shared by several simulations. If this attribute is nil, the link can only be used locally.

master_obj_id

Pseudo attribute; **read/write** access; type: `integer`. The local ID number of the master link object on the link.

min_latency

Pseudo attribute; **read/write** access; type: `integer`. The minimum latency allowed. This is an integer number of clock ticks in the time base specified in the *frequency* attribute. In a distributed the value is given by Simics Central server. This attribute can not be changed.

throttle

Optional attribute; **read/write** access; type: `integer`. The maximum number of bits per second that may be sent over the link in each direction. A value of zero disables throttling

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<serial-link>.info

Synopsis

<serial-link>.info

Description

Print detailed information about the configuration of the device.

<serial-link>.status

Synopsis

<serial-link>.status

Description

Print detailed information about the current status of the device.

service-node

Provided By
[service-node](#)

Interfaces Implemented
conf_object, [log_object](#)

Description

This class provides services on an Ethernet network.

Attributes

allow_real_dns

Optional attribute; **read/write** access; type: `boolean`. If TRUE, the service node's DNS server will forward queries for unknown hosts to a real DNS server.

bootparam_udp_port

Optional attribute; **read/write** access; type: `integer`. The UDP port currently used for the Sun RPC Bootparam protocol replies. For checkpointing only.

default_boot_filename

Optional attribute; **read/write** access; type: `string` or `nil`. Default boot filename supplied in replies from the BOOTP/DHCP server if no client specific filename is configured in the host_boot_filename attribute.

default_dhcp_options

Optional attribute; **read/write** access; type: `[[id]*]`. Extension of the build-in set of DHCP options that the DHCP server uses in replies. These default options are used if no client specific options are set in the host_dhcp_options attribute.

dhcp6_allocated_addresses

Optional attribute; **read/write** access; type: `[[sii]*]`. The allocated addresses. The sub-entries for each allocated address are *address*, *commit-flag*, *decline-flag*. *address* must be an IPv6 address.

dhcp6_clients

Optional attribute; **read/write** access; type: `[[d[[iii[[sii]*]]*][[i[[sii]*]*]*]*]`. All known clients.

dhcp6_dns_servers

Optional attribute; **read/write** access; type: `[s*]`. List of IPv6 addresses to DNS recursive name servers.

dhcp6_domain_search_list

Optional attribute; **read/write** access; type: `[s*]`. List of domain names representing the domain search list.

dhcp6_lifetimes

Optional attribute; **read/write** access; type: [ii]. Preferred and valid lifetimes for IPv6 addresses.

dhcp6_nis_domain_name

Optional attribute; **read/write** access; type: string. The NIS domain name.

dhcp6_nis_servers

Optional attribute; **read/write** access; type: [s*]. List of IPv6 addresses to NIS servers.

dhcp6_nisp_domain_name

Optional attribute; **read/write** access; type: string. The NIS+ domain name.

dhcp6_nisp_servers

Optional attribute; **read/write** access; type: [s*]. List of IPv6 addresses to NIS+ servers.

dhcp6_serverid

Optional attribute; **read/write** access; type: data or string. The DUID identifying the DHCPv6 server.

dhcp6_valid_prefixes

Optional attribute; **read/write** access; type: [[is]*]. The IPv6 prefixes that are valid on the local link. The sub-entries for each prefix are *prefix-length*, *ip-address*. *ip-address* must be an IPv6 address.

dhcp_leases

Optional attribute; **read/write** access; type: [[ssdi]*]. The active leases used by the DHCP service. The sub-entries for each lease are *ip-address*, *mac-address*, *client-id*, *lease-time*, *timestamp*. The lease time is given in seconds since the lease was issued.

dhcp_max_lease_time

Optional attribute; **read/write** access; type: integer or nil. The maximum lease time handed out by the DHCP server. If the value is Nil, or 0xffffffff, infinite leases are allowed.

eth_interfaces

Pseudo attribute; **read-only** access; type: [o*]. The ethernet interface objects used by this service node.

host_boot_filename

Optional attribute; **read/write** access; type: [[ss|n]*]. Extensions to the host database with the BOOTP/DHCP boot filename. Each entry in the list is a sub list with a client IP address and the boot filename. If no entry exists in the host database for a client boots using BOOTP or DHCP, then the value of the default_boot_filename attribute is used.

host_dhcp_options

Optional attribute; **read/write** access; type: `[[s[[id]*]]*]`. Extensions to the host database with client specific DHCP option values that the DHCP server uses in replies. If no client specific option exists, then the value in the `default_dhcp_options` attribute, if set, is used. There are also a number of built-in DHCP options that the server always replies to. A built-in option can be disabled by specifying an empty value for the option in question. Each entry in the attribute list is a sub list with a client IP address followed by yet one list with two entries: the option number and a byte array with the reply. Vendor specific options can not be defined using this attribute.

host_pools

Optional attribute; **read/write** access; type: `[[i,s,s,s[[s,s,s,s]*]]*]`. The host database used by the DHCP and DNS servers.

hosts

Optional attribute; **read/write** access; type: `[[s|n,s,s,s]*]`. The host database used by the DHCP and DNS servers.

napt_enable

Optional attribute; **read/write** access; type: `integer`. When set to non-zero, the service node will provide network address port translation as specified by RFC 3022. The current implementation can only handle TCP NAPT.

recorder

Required attribute; **read/write** access; type: `object`. Recorder device for recording and playback of network traffic.

routing_table

Optional attribute; **read/write** access; type: `[[ss|iso]*]`. The routing table used by this service node. The four sub-entries for each routing table entry are *network*, *netmask*, *gateway* and *service-node*. The network and gateway are specified as IP addresses in dot notation, and the netmask is specified as an IP address or as a prefix length. The service-node is the node connected to the network where the gateway resides. For a route where the destination is reachable on the other service-node's connected network, the gateway should be set to 0.0.0.0. To create a default route, both the network and netmask should be set to 0.0.0.0.

rpc_bindings

Optional attribute; **read/write** access; type: `[[iiii]*]`. List of RPC bindings (program, version, protocol, port) that are used by port mapper or RPCBIND services within the service node.

services

Optional attribute; **read/write** access; type: `dictionary`. The network services provided by the service-node, as a dictionary, and their enabled/disabled status.

tcp_pcbs

tcp_pcbs_all

tftp_sessions

Optional attribute; **read/write** access; type: `[[iissssi]*]`. The active sessions in the TFTP service. The sub-entries for each session are *server TID*, *client TID*, *read/write*, *server IP address*, *client IP address*, *file name*, *last read block*.

udp_pcbs

Optional attribute; **read/write** access; type: `[[o|ni|si|siiiiioi] *]`. UDP PCBs.
Do not attempt to setup connections yourself using this attribute, but use functionality
in each separate UDP service to do so.

udp_pcbs_all

Pseudo attribute; read-only access; type: `[[o|ni|si|siiiiiii|oi]*]`. UDP PCBs. This attribute is identical to `udp_pcbs` with the exception that it can only be read and that it also includes connections that will not be checkpointed.

Command List

Commands

add-host	add host entry
arp	inspect and manipulate ARP table
connect	<i>deprecated</i> — connect to an ethernet link
delete-host	delete host entry
dhcp-add-pool	add DHCP pool
dhcp-leases	show DHCP leases
disable-real-dns	disable real DNS
disable-service	disable network service
enable-ftp-alg	enable FTP ALG
enable-real-dns	enable real DNS
enable-service	enable network service
info	print information about the device
list-host-info	print host info database
route	show the routing table
route-add	add an entry to the routing table
set-tftp-directory	set TFTP directory

status print status of the device
tcpip-info show TCP/IP info

Command Descriptions

<service-node>.add-host

Synopsis

<service-node>.add-host “*ip*” “*name*” [“*domain*”] [“*mac*”]

Description

Add a host entry to the DHCP and DNS server tables.

See Also

[`<service-node>.list-host-info`](#)

<service-node>.arp

Synopsis

<service-node>.arp [-d] [“*delete-ip*”]

Description

Prints the ARP table for the service node if called with no arguments. An ARP entry can be deleted using the -d flag and the IP address.

<service-node>.connect — *deprecated*

Synopsis

<service-node>.connect *link* “*ip*” [“*netmask*”] [*vlan*]

Description

This command is deprecated; use [`<std-service-node>.connect-to-link`](#) instead.

Connect this service node to an Ethernet link object.

The *ip* argument gives the IP address that the service node will use on the link, and the *netmask* argument the netmask. Optionally, the netmask may be given in the *ip* argument as a /*bits* suffix. The netmask may also left out entirely, in which case it will default to 255.255.255.0.

<service-node>.delete-host

Synopsis

<service-node>.delete-host [“*ip*”] [“*name*”] [“*domain*”] [“*mac*”]

Description

Delete a host entry from the DHCP and DNS server tables.

See Also

[`<service-node>.list-host-info`](#)

`<service-node>.dhcp-add-pool`

Synopsis

`<service-node>.dhcp-add-pool pool-size "ip" ["name"] ["domain"]`

Description

Add an IP address pool to the DHCP server. The *pool-size* parameter defines the number of available addresses in the pool, starting with address *ip*. The DNS server will map the addresses to a name that is the *name* parameter with a number appended, in the *domain* domain.

`<service-node>.dhcp-leases`

Synopsis

`<service-node>.dhcp-leases`

Description

Print the list of active DHCP leases.

`<service-node>.disable-real-dns`

Synopsis

`<service-node>.disable-real-dns`

Description

Disable forwarding of DNS queries for unknown hosts to the real DNS server.

`<service-node>.disable-service`

Synopsis

`<service-node>.disable-service ["name"] [-all]`

Description

Disable a named network service in the service-node, or all of the *-all* flag is used.

`<service-node>.enable-ftp-alg`

Synopsis

`<service-node>.enable-ftp-alg`

Description

Enable the FTP ALG. FTP ALG processing is needed to support port forwarded FTP from the outside network to simulated machines.

<service-node>.enable-real-dns

Synopsis

```
<service-node>.enable-real-dns
```

Description

Enable forwarding of DNS queries for unknown hosts to the real DNS server.

<service-node>.enable-service

Synopsis

```
<service-node>.enable-service ["name"] [-all]
```

Description

Enable a named network service in the service-node, or all of the *-all* flag is used.

<service-node>.info

Synopsis

```
<service-node>.info
```

Description

Print detailed information about the configuration of the device.

<service-node>.list-host-info

Synopsis

```
<service-node>.list-host-info
```

Description

Print the host information database, used by the DHCP and DNS server.

See Also

[`<service-node>.add-host`](#)

<service-node>.route

Synopsis

```
<service-node>.route [-port]
```

Description

Print the routing table.

By default, the name of the link on which traffic will be send is printed in the last column, but if the *-port* flag is use, the port device will be printed instead.

<service-node>.route-add

Synopsis

```
<service-node>.route-add "net" ["netmask"] ["gateway"] link
```

Description

Add to the routing table.

<service-node>.set-tftp-directory**Synopsis**

```
<service-node>.set-tftp-directory (dir|-default)
```

Description

Set the directory used by the TFTP service for files read and written over TFTP. By default, the *SIM_lookup_file* is used for finding files to read, and the current directory is used for writing files.

<service-node>.status**Synopsis**

```
<service-node>.status
```

Description

Print detailed information about the current status of the device.

<service-node>.tcpip-info**Synopsis**

```
<service-node>.tcpip-info
```

Description

Print all TCP/IP connections.

service-node-device

Provided By
[service-node](#)

Interfaces Implemented
conf_object, [ethernet_common](#), [ethernet_device](#), [log_object](#)

Description
Ethernet device for the service-node class.

Attributes

ip_addresses

Optional attribute; **read/write** access; type: [s*]. The configured IP addresses for this network interface. They are on the form "*addr/prefixlen*" and allows both IPv4 and IPv6 addresses. This list does not include automatically assigned IP number, such as link-local IPv6 addresses.

link

Pseudo attribute; **read/write** access; type: object or nil. The network link that this service node ethernet device is attached to.

link_state

Optional attribute; **read/write** access; type: object or nil. The network link that this service node ethernet device is attached to (for checkpointing).

mac_address

Optional attribute; **read/write** access; type: [iiiiii] or string. The MAC address used by this service node ethernet device.

mtu

Optional attribute; **read/write** access; type: integer. MTU, maximum transfer unit, is the largest Ethernet payload (without Ethernet header and CRC) that will be sent out on the network without fragmentation. Default 1500.

neighbors

Optional attribute; **read/write** access; type: dictionary. The neighbor table used by this ethernet interface.

service_node

Required attribute; **read/write** access; type: object. The service node that the ethernet device is attached to.

tx_bandwidth

Optional attribute; **read/write** access; type: integer. The transmit bandwidth of the network interface in bits per second. The network interface will limit the rate at which it sends packets to remain below this bandwidth. Default limit is 1Gbit/s.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<service-node-device>.info

Synopsis

<service-node-device>.info

Description

Print detailed information about the configuration of the device.

<service-node-device>.status

Synopsis

<service-node-device>.status

Description

Print detailed information about the current status of the device.

set-memory

Provided By

[set-memory](#)

Interfaces Implemented

[conf_object](#), [io_memory](#), [log_object](#)

Description

The set-memory device is a device used to represent memory that is all set to a specific value. For example unmapped bios area in a PC that should return -1 (all ones).

Attributes

value

Optional attribute; **read/write** access; type: `integer`. The byte value returned when the memory is accessed.

Command List

Commands

[info](#) print information about the device

[status](#) print status of the device

Command Descriptions

<set-memory>.info

Synopsis

`<set-memory>.info`

Description

Print detailed information about the configuration of the device.

<set-memory>.status

Synopsis

`<set-memory>.status`

Description

Print detailed information about the current status of the device.

signal-bus

Provided By
[signal-bus](#)

Interfaces Implemented
conf_object, [log_object](#), signal

Description

The **signal-bus** class implements a signal bus that distributes a signal from one or more initiators to a configured list of targets. Both the signal bus itself and all targets implement the `signal` interface.

Attributes

input_inverted

Optional attribute; **read/write** access; type: boolean. True if the input signal should be inverted before sent out to the connected targets. Default is False.

targets

Optional attribute; **read/write** access; type: `[o| [o, n|s] | [o, n|s, b]*]`. Signal target objects, implementing one or more instances of the `signal` interface. The third argument in the target sub-list can be set to True if the target should get the signal inverted.

Command List

Commands

[info](#) print information about the device
[status](#) print status of the device

Command Descriptions

<signal-bus>.info

Synopsis

<signal-bus>.info

Description

Print detailed information about the configuration of the device.

<signal-bus>.status

Synopsis

<signal-bus>.status

Description

Print detailed information about the current status of the device.

signal_link_endpoint

Provided By

[signal-link](#)

Interfaces Implemented

[conf_object](#), [log_object](#), [signal](#)

Description

Signal link endpoint

Attributes

device

Required attribute; **read/write** access; type: `[os], object, or nil`. The device connected to this endpoint.

id

Required attribute; **read/write** access; type: `integer`. Endpoint ID. The ID of each endpoint must be unique among the link's endpoints, and it may not be 0 or `0xffffffffffffffff`

link

Required attribute; **read/write** access; type: `object`. The link object to which this endpoint belongs.

type

Required attribute; **read/write** access; type: `string`. Endpoint type ("sender" or "receiver").

Command List

Commands

[**info**](#) print information about the device

[**status**](#) print status of the device

Command Descriptions

`<signal_link_endpoint>.info`

Synopsis

`<signal_link_endpoint>.info`

Description

Print detailed information about the configuration of the device.

`<signal_link_endpoint>.status`

Synopsis

<signal_link_endpoint>.status

Description

Print detailed information about the current status of the device.

signal_link_impl

Provided By

[signal-link](#)

Interfaces Implemented

`conf_object`, [log_object](#)

Description

A link that propagates simple signals.

Attributes

goal_latency

Required attribute; **read/write** access; type: `float`. The desired latency for this link.

Command List

Commands

info print information about the device

status print status of the device

Command Descriptions

<signal_link_impl>.info

Synopsis

`<signal_link_impl>.info`

Description

Print detailed information about the configuration of the device.

<signal_link_impl>.status

Synopsis

`<signal_link_impl>.status`

Description

Print detailed information about the current status of the device.

signal_to_interrupt

Provided By

[signal-to-interrupt](#)

Interfaces Implemented

conf_object, [log_object](#), signal

Description

When the signal is raised on an instance of this class the device connected to irq_dev will be interrupted on level irq_level. Similarly, when the signal is lowered the interrupt is cleared.

Attributes

irq_dev

Required attribute; **read/write** access; type: [os] or object. When the signal of an instance of the signal_to_interrupt class is raised the irq_dev device will receive an interrupt with level specified by the irq_level attribute.

irq_level

Required attribute; **read/write** access; type: integer. When the signal of an instance of the signal_to_interrupt class is raised the device specified by the irq_dev attribute will receive an interrupt at level irq_level

Command List

Commands

[info](#) print information about the device

[status](#) print status of the device

Command Descriptions

<signal_to_interrupt>.info

Synopsis

<signal_to_interrupt>.info

Description

Print detailed information about the configuration of the device.

<signal_to_interrupt>.status

Synopsis

<signal_to_interrupt>.status

Description

Print detailed information about the current status of the device.

SIL680A

Provided By

[sil680a](#)

Interfaces Implemented

[conf_object](#), [io_memory](#), [log_object](#), [simple_interrupt](#)

Description

Bus master IDE function from SiL680A.

Command List

Commands

info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
status	print status of the device

Command Descriptions

<SIL680A>.info

Synopsis

`<SIL680A>.info`

Description

Print detailed information about the configuration of the device.

<SIL680A>.pci-header — *deprecated*

Synopsis

`<SIL680A>.pci-header [-v]`

Description

This command is deprecated; use [<SIL680A>.print-pci-config-reg](#)s instead.

<SIL680A>.print-pci-config-reg

Synopsis

`<SIL680A>.print-pci-config-reg [-v]`

Description

Print the PCI configuration registers. The *-v* flag turns on verbose mode.

<SIL680A>.status

Synopsis

<SIL680A>.status

Description

Print detailed information about the current status of the device.

sim

Provided By

[Simics Core](#)

Interfaces Implemented

`conf_object`, [log_object](#)

Description

The **sim** class represent Simics's global simulator state, such as state not tied to a specific processor, device etc.

Attributes

automatic_cell_partition

Session attribute; **read/write** access; type: `boolean`. When TRUE, standard Simics components will automatically create cells as appropriate during the instantiation of models. When set to FALSE, all objects instantiated afterwards will be assigned to a default cell, providing a timing behavior compatible with previous Simics versions (3.2 and before).

checkpoint_path

Optional attribute; **read/write** access; type: `[s*]`; **persistent** attribute. File search path for previous checkpoints, used by the `files` attribute in the **image** class.

command_log

Session attribute; **read/write** access; type: `boolean`. Set to TRUE (default) if command should be logged to the command log file. See also the `prefs->command_log` attribute. This attribute may differ from the preference value if the `-log` or `-no-log` flag was specified at startup.

cpu_mode

Pseudo attribute; **read/write** access; type: `string`. The current CPU mode used when loading processor modules. One of normal and stall. See also the `prefs->cpu_mode` attribute. This attribute may differ from the preference value if one of the `-stall` and `-fast` flags was specified at startup.

default_log_level

Optional attribute; **read/write** access; type: `integer`. The log level used for newly created objects.

deprecation_level

Session attribute; **read/write** access; type: `integer`. A level of 0 turns off all deprecation warnings. On level 1 (default) Simics will warn when features that have been deprecated one major release back or longer are used. On level 2, Simics will warn when newly deprecated features, or features that will become deprecated, are used.

deprecation_stack_trace

Session attribute; **read/write** access; type: boolean. Set to true to make Simics print a stack trace each time a deprecated feature is used.

fail_on_warnings

Session attribute; **read/write** access; type: boolean. Set to TRUE (default is FALSE) if Simics should treat some warnings as errors.

gui_mode

Pseudo attribute; **read/write** access; type: string. The current GUI mode. One of “gui”, “mixed” and “no-gui”. See also the prefs->gui_mode attribute. This attribute may differ from the preference value if one of the -gui, -gui-only and -no-win flags was specified at startup.

handle_outside_memory

Optional attribute; **read/write** access; type: boolean. When a physical address is accessed which is neither memory (RAM) nor a memory-mapped device, Simics will default to stop on a special breakpoint and report this, on the assumption that it is erroneous behavior. If this attribute is set to TRUE, then Simics will override the default behavior and instead handle the operation in the manner prescribed by the target architecture. For example, this might trigger a data access exception.

hide_console_windows

Session attribute; **read/write** access; type: boolean. A flag indicating whether Simics should hide windows for consoles. Default FALSE.

host_arch

Pseudo attribute; **read-only** access; type: string. Host architecture. *x86* or *amd64*.

host_bits

Pseudo attribute; **read-only** access; type: string. Host bits. 32 or 64.

host_num_cpus

Pseudo attribute; **read-only** access; type: integer. The number of processors on the host.

host_os

Pseudo attribute; **read-only** access; type: string. Host operating system. *linux* or *windows*.

host_phys_mem

Pseudo attribute; **read-only** access; type: integer. The amount of physical memory of the host in bytes.

host_type

Pseudo attribute; **read-only** access; type: string. Host type. *linux32*, *linux64*, *win32* or *win64*.

license_file

Pseudo attribute; **read-only** access; type: `string` or `nil`. The currently used FlexNET license-file. See also the `prefs->license_file` attribute. This attribute may differ from the preference value if an alternative license file was specified at startup.

multithread_warnings

Session attribute; **read/write** access; type: `boolean`. Control the warnings for non-thread-safe operations when running Simics in single thread mode. If false, no warnings are shown. If true, Simics will print warnings about non-thread-safe operations (Default is false). If running Simics in multithread mode, warnings are always activated and may be fatal, depending on how serious the problem is.

package_list

Pseudo attribute; **read-only** access; type: `string` or `nil`. Package list file used in this session (if any).

simics_base

Pseudo attribute; **read-only** access; type: `string`. Full path to the Simics base installation directory.

simics_home

Pseudo attribute; **read-only** access; type: `string`. Full path to the Simics host directory.

simics_path

Optional attribute; **read/write** access; type: `[s*]`. The value of the Simics search path. See the commands `add-directory` and `lookup-file` for additional information. The current directory is always searched before any entry in the Simics search path.

timing_model_inquiries

Session attribute; **read/write** access; type: `boolean`. If set to TRUE, then timing models will receive inquiries in addition to real memory operations. Default is FALSE.

version

Optional attribute; **read/write** access; type: `integer`. Version (build-id) of Simics that a configuration file was created with.

workspace

Pseudo attribute; **read/write** access; type: `string` or `nil`. Full path to the current workspace directory. Returns NIL if no workspace is used. This attribute should not be written. See also the `prefs->workspace` attribute. This attribute may differ from the preference value if -workspace was specified at startup.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<sim>.info

Synopsis

<sim>.info

Description

Print detailed information about the configuration of the device.

<sim>.status

Synopsis

<sim>.status

Description

Print detailed information about the current status of the device.

simple-scsi-disk

Provided By

[simple-scsi-disk](#)

Interfaces Implemented

`conf_object`, [log_object](#)

Description

Simple model of a SCSI disk for use with a SCSI controller. This model is not intended to be connected to a SCSI bus.

Attributes

controller

Required attribute; **read/write** access; type: `object`. Name of the controller that this disk is connected to.

geometry

Required attribute; **read/write** access; type: `[iii]`. The geometry of the disk. ()

image

Required attribute; **read/write** access; type: `object`. Name of the image object holding the actual data of the disk. This object must implement the 'image' interface.

Command List

Commands

add-diff-file	add a diff file to the image
add-diff-partial-file	add a partial diff file to the image
add-sun-partition	add partition from a file
create-partition	add a partition to disk
create-sun-vtoc-header	write a new VTOC to a Sun disk
create-sun-vtoc-partition	write partition data in the VTOC on a Sun disk
delete-sun-vtoc-partition	delete partition data from the VTOC on a Sun disk
dump-sun-partition	write partition as a file
info	information about current state of the SCSI disk
print-partition-info	print info about a pc-style partition
print-partition-table	print the disk's pc-style partition table
print-sun-vtoc	print the VTOC for a Sun disk
save-diff-file	save diff file to disk

Command Descriptions

`<simple-scsi-disk>.add-diff-file`

Synopsis

`<simple-scsi-disk>.add-diff-file filename [-replace] [-rw]`

Description

Add a diff file to the list of files for an disk. The diff file was typically created with the 'save-diff-file' command, or by a saved configuration. This is basically the same command as <image>.add-diff-file. The file can be made writable instead of read-only using the -rw flag. To replace any existing files, use -replace.

See Also

[`<simple-scsi-disk>.save-diff-file`](#), [`<image>.add-diff-file`](#)

`<simple-scsi-disk>.add-diff-partial-file`

Synopsis

`<simple-scsi-disk>.add-diff-partial-file filename start [size]`

Description

Add a diff file to the list of files for an disk. The diff file was typically created with the 'save-diff-file' command, by one of the dump-* partition commands, or by a saved configuration. This is basically the same command as <image>.add-partial-diff-file.

See Also

[`<simple-scsi-disk>.save-diff-file`](#), [`<image>.add-diff-file`](#), [`<image>.add-partial-diff-file`](#)

`<simple-scsi-disk>.add-sun-partition`

Synopsis

`<simple-scsi-disk>.add-sun-partition number file`

Description

Adds an image or diff as a sun partition to the current disk.

See Also

[`<simple-scsi-disk>.dump-sun-partition`](#)

`<simple-scsi-disk>.create-partition`

Synopsis

`<simple-scsi-disk>.create-partition partition start_cylinder start_head start_sector end_cylinder end_head end_sector (type_id|"type_name") start_lba size_in_sectors [-boot-indication]`

Description

Add a partition to a disk's partition table. Will create a primary PC-style partition with number *partition*. Any existing partition will be overwritten. The start location of the partition is given both in CHS format with *start_cylinder*, *start_head*, *start_sector*, and in linear format with *start_lba*. The partition endpoint is specified in CHS format with *end_cylinder*, *end_head*, and *end_sector*. The size in 512-byte sectors is *size_in_sectors*.

The slight overspecification of the parameters is there because they depend on how the BIOS translates the disk geometry to the BIOS geometry. All CHS coordinates given to this command should be in the translated BIOS geometry.

The partition boot bit is enabled with the *-boot-indication* flag (default off).

The partition type can either raw numerical code, or a symbolic name. The following symbolic names are understood by the command: "unused" (0x0), "DOS 12-bit FAT" (0x1), "DOS 3.0+ 16-bit FAT" (0x4), "DOS 3.3+ extended" (0x5), "DOS 3.31+ 16-bit FAT" (0x6), "OS/2 IFS" (0x7), "OS/2 boot manager" (0xa), "Win95 32-bit FAT" (0xb), "Win95 32-bit FAT, LBA" (0xc), "Win95 16-bit FAT, LBA" (0xe), "Win95 extended, LBA" (0xf), "Linux swap" (0x82), "Linux native" (0x83), "Linux extended" (0x85), "Linux LVM" (0x8e), "BSD/386" (0xa5), "OpenBSD" (0xa6), "NetBSD" (0xa9), and "BeOS" (0xeb).

See Also

[`<simple-scsi-disk>.print-partition-table`](#)

`<simple-scsi-disk>.create-sun-vtoc-header`

Synopsis

`<simple-scsi-disk>.create-sun-vtoc-header [C] [H] [S] [-quiet]`

Description

Create and write a new VTOC to a Sun disk. The geometry information written is taken from the configuration attribute 'geometry' of the disk, unless specified with the **C**, **H** and **S** parameters. A new empty partition table is also created, with only the standard 'backup' partition as number 2. *-quiet* makes the command silent in case of success.

See Also

[`<simple-scsi-disk>.print-sun-vtoc`](#), [`<simple-scsi-disk>.create-sun-vtoc-partition`](#), [`<simple-scsi-disk>.delete-sun-vtoc-partition`](#)

`<simple-scsi-disk>.create-sun-vtoc-partition`

Synopsis

`<simple-scsi-disk>.create-sun-vtoc-partition number "tag" "flag" start-block num-blocks [-quiet]`

Description

Write partition information to the VTOC on a Sun disk. This command does not change the format of the disk, and it does not create any file system on the partition. Only the 'Volume Table Of Contents' is modified. No checking is performed to make sure that partitions do not overlap, or that they do not exceed the disk size. *-quiet* makes the command silent in case of success.

See Also

[`<simple-scsi-disk>.print-sun-vtoc`](#), [`<simple-scsi-disk>.create-sun-vtoc-header`](#), [`<simple-scsi-disk>.delete-sun-vtoc-partition`](#)

<simple-scsi-disk>.delete-sun-vtoc-partition

Synopsis

<simple-scsi-disk>.delete-sun-vtoc-partition *number* [-quiet]

Description

Delete the information in the VTOC on a Sun disk for the specified partition. No other modification on the disk is performed. *-quiet* makes the command silent in case of success.

See Also

[**<simple-scsi-disk>.print-sun-vtoc**](#), [**<simple-scsi-disk>.create-sun-vtoc-header**](#), [**<simple-scsi-disk>.create-sun-vtoc-partition**](#)

<simple-scsi-disk>.dump-sun-partition

Synopsis

<simple-scsi-disk>.dump-sun-partition *number file*

Description

Write all data from a Sun disk partition to the specified file in raw format.

See Also

[**<simple-scsi-disk>.print-sun-vtoc**](#), [**<simple-scsi-disk>.add-sun-partition**](#)

<simple-scsi-disk>.info

Synopsis

<simple-scsi-disk>.info

Description

Print information about the state of the simple SCSI disk.

<simple-scsi-disk>.print-partition-info

Synopsis

<simple-scsi-disk>.print-partition-info *num [filename]*

Description

Print information about a partition. If a filename is given, then suitable partition parameters for use with image aware e2tools will be written to that file.

<simple-scsi-disk>.print-partition-table

Synopsis

<simple-scsi-disk>.print-partition-table

Description

Print the PC-style partition table for a disk.

See Also

[`<simple-scsi-disk>.create-partition`](#)

<simple-scsi-disk>.print-sun-vtoc

Synopsis

`<simple-scsi-disk>.print-sun-vtoc`

Description

Print the contents of the VTOC (volume table of contents) for a Sun disk. This is similar to the Solaris 'prtvtoc' command.

See Also

[`<simple-scsi-disk>.create-sun-vtoc-header`](#), [`<simple-scsi-disk>.create-sun-vtoc-partition`](#),
[`<simple-scsi-disk>.delete-sun-vtoc-partition`](#)

<simple-scsi-disk>.save-diff-file

Synopsis

`<simple-scsi-disk>.save-diff-file filename`

Description

Writes changes to the image as a diff file in craff format. This is basically the same command as `<image>.save-diff-file`.

See Also

[`<simple-scsi-disk>.add-diff-file`](#), [`<image>.save-diff-file`](#)

slaver

Provided By

[Simics Core](#)

Interfaces Implemented

[conf_object](#), [log_object](#), [slaver_message](#)

Description

System for synchronizing with external processes.

Attributes

agent

Required attribute; **read/write** access; type: `object`. Object handling messages to/from the slave.

clock

Required attribute; **read/write** access; type: `object`. Proxy clock for the slave.

Command List

Commands

[**info**](#) print information about the device
[**status**](#) print status of the device

Command Descriptions

<slaver>.info

Synopsis

`<slaver>.info`

Description

Print detailed information about the configuration of the device.

<slaver>.status

Synopsis

`<slaver>.status`

Description

Print detailed information about the current status of the device.

software_tracker

Provided By
[software-tracker](#)

Interfaces Implemented
[conf_object](#), [log_object](#), [software](#)

Description

(Internal) Base class that is utilized by Simics OS-awareness.

Attributes

processors

Required attribute; **read/write** access; type: [o+]. The (full) set of processors running the software to be tracked. May not be modified once the software tracker has been created.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<software_tracker>.info

Synopsis

<software_tracker>.info

Description

Print detailed information about the configuration of the device.

<software_tracker>.status

Synopsis

<software_tracker>.status

Description

Print detailed information about the current status of the device.

state-assertion

Provided By
[state-assertion](#)

Interfaces Implemented

`conf_object`, [log_object](#), [snoop_memory](#), [timing_model](#)

Description

The *state-assertion* extension provides the possibility to record the state of various devices at regular interval, to be able to compare it to a later run.

When creating trace assertion information, the general scheme is the following:

- use state-assertion-create-file to open the file where the data will be saved. This will also create a state-assertion object.
- use `<state-assertion>.add` to specify the objects to assert as well as the amount of steps between each assertion.
- use `<state-assertion>.start` to begin the assertion. From now on, continuing the simulation will record information into the file.
- use `<state-assertion>.stop` to empty the buffers and close the file.

When using state-assertion information to compare against the current run, you should:

- use state-assertion-open-file to open the file containing the state information you want to use.
- use `<state-assertion>.start` when you want to begin the comparison. The comparison is stopped automatically when no data is available. If states differ, the simulation is stopped and the differences are displayed.

It is also possible to connect directly two Simics instead of using a file, with the functions state-assertion-receive and state-assertion-connect. The recording/comparison is then executed directly.

Command List

Commands

add	add an object to be asserted
add-mem-lis	add a memory listener on the specified memory space
ffoward	fast-forward a state assertion file when comparing
info	provide information about the state assertion
start	start trace asserting/comparing
status	provide the status of the current state assertion
stop	stop trace asserting/comparing and close the file

Command Descriptions

`<state-assertion>.add`

Synopsis

`<state-assertion>.add obj steps [type] ["attribute"] [step-obj]`

Description

Add an object to a state assertion file so its state will be recorded.

- *obj* is the name of the object to be added.
- *steps* is the number of steps between each save.
- *type* is the type of state saved in the file (for devices that provide several, the most complete state is saved by default).
- *attribute* is the attribute to save. If specified, the save_state interface is not used and the attribute is saved instead. This is useful for object having no save_state interface.
- *step-obj* is the object that implements the event queue you want to post the event to. If not specified, the obj must implement step interface.

<state-assertion>.add-mem-lis

Synopsis

```
<state-assertion>.add-mem-lis "memory_space"
```

Description

Add a memory listener to a memory space so that all memory transactions will be recorded in the file.

- *memory_space* is the name of the memory space to listen to.
- *-o* allows state-assertion to take over an existing memory hierarchy.

<state-assertion>.fforward

Synopsis

```
<state-assertion>.fforward arg1 steps
```

Description

Fast-forward a state assertion file. The contents of the file are ignored until the object *obj* has skipped *steps* steps. The simulation is not fast-forwarded. Other objects in the file are fast-forwarded along.

<state-assertion>.info

Synopsis

```
<state-assertion>.info
```

Description

Describe the state assertion performed by the current object.

<state-assertion>.start

Synopsis

```
<state-assertion>.start
```

Description

Start the recording/comparison.

```
<state-assertion>.status
```

Synopsis

```
<state-assertion>.status
```

Description

Describe the status of the state assertion performed by the current object.

```
<state-assertion>.stop
```

Synopsis

```
<state-assertion>.stop
```

Description

Stop the recording/comparison, flush the buffers and close the file.

static_link_connector

Provided By

[Simics Core](#)

Interfaces Implemented

[conf_object](#), [connector](#), [log_object](#)

Description

Static link connector representing one connection to the link. This connector does not grow dynamically and can be reused several times. It is useful for links that have specific ports to which only one device can connect (for example, a serial link with two devices).

Attributes

connector_name

Required attribute; **read/write** access; type: `string`. Name of the connector

connector_template

Required attribute; **read/write** access; type: `string`. Name of the link connector template used to create this connector

connector_type

Required attribute; **read/write** access; type: `string`. Type used to match which other connector objects this connector can connect to.

direction

Required attribute; **read/write** access; type: `integer`. Direction of the connector: up, down or any

hotpluggable

Required attribute; **read/write** access; type: `boolean`. If true, this connector can be connected or disconnected after instantiation. If false, the connection must be made before the component is instantiated.

multi

Required attribute; **read/write** access; type: `boolean`. If true, more than one connector object can be connected to this connector at the same time.

owner

Required attribute; **read/write** access; type: `object`. Component to which this connector applies

required

Required attribute; **read/write** access; type: `boolean`. If true, this connector should be connected before its component can be instantiated. If false, it can be left empty.

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<static_link_connector>.info

Synopsis

<static_link_connector>.info

Description

Print detailed information about the configuration of the device.

<static_link_connector>.status

Synopsis

<static_link_connector>.status

Description

Print detailed information about the current status of the device.

status_panel

Provided By
[status-panel](#)

Interfaces Implemented
conf_object, [log_object](#)

Description

Unsupported class, do not use! The **status_panel** class is used to model simple control and status panels with items such as buttons and LEDs. One or several **status_panels** can be groups together using a **status_panel_view** object and viewed in a GUI window. There are also a CLI command for generating input events.

Attributes

device

Optional attribute; **read/write** access; type: `object` or `nil`. A device object that receives any key events. This object must implement the `keypad` interface.

items

Required attribute; **read/write** access; type: `[[D*] *]`. Control and display items of the panel. The format is a list with one list per row. Each item is a dictionary with the following keys:

panel_view

Optional attribute; **read/write** access; type: `object` or `nil`. The **status_panel_view** object used to view this panel, possibly together with other ones, in a GUI window.

recorder

Required attribute; **read/write** access; type: `object`. Recorder object.

title

Optional attribute; **read/write** access; type: `string`. Title of the status panel.

user_type

Optional attribute; **read/write** access; type: `integer`. User defined type of the panel. This value can be used by the corresponding `panel_view` to determine how different panels are laid out in relation to each other.

status_panel_view

Provided By

[status-panel-view](#)

Interfaces Implemented

conf_object, [log_object](#)

Description

Unsupported class, do not use! The **status_panel_view** class is used to group several **status_panel** objects together in a single view. If the Simics GUI is available, a window with the panels and their controls is opened.

Attributes

gui

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE if a GUI window with the panel view should be opened, and FALSE if not. Default is to open a window.

title

Optional attribute; **read/write** access; type: `string`. Title of the panel view window.

SYM53C810

Provided By
[SYM53C810](#)

Interfaces Implemented
`conf_object`, [io_memory](#), [log_object](#)

Description

The SYM53C810 is an advanced SCSI controller with integrated DMA capabilities. The SYM53C810 device does not support target mode operation and can only act as initiator during SCSI transactions. The SYM53C810 instruction set implementation is not complete and foremost load/store instructions are missing. Assembly language instructions shown with debug level greater than 1 do not comply with the Symbios Logic assembler format.

Attributes

scsi_bus

Required attribute; **read/write** access; type: `object`. Object must implement the 'scsi-bus' interface.

Command List

Commands

info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
status	print status of the device

Command Descriptions

<SYM53C810>.info

Synopsis

`<SYM53C810>.info`

Description

Print detailed information about the configuration of the device.

<SYM53C810>.pci-header — *deprecated*

Synopsis

`<SYM53C810>.pci-header [-v]`

Description

This command is deprecated; use [<SYM53C810>.print-pci-config-reg](#)s instead.

<SYM53C810>.print-pci-config-reg

Synopsis

<SYM53C810>.print-pci-config-reg [-v]

Description

Print the PCI configuration registers. The *-v* flag turns on verbose mode.

<SYM53C810>.status

Synopsis

<SYM53C810>.status

Description

Print detailed information about the current status of the device.

SYM53C875

Provided By
[SYM53C875](#)

Interfaces Implemented
`conf_object`, [io_memory](#), [log_object](#)

Description

The SYM53C875 is an advanced SCSI controller with integrated DMA capabilities and dedicated script RAM to decrease bus traffic. The SYM53C875 device does not support target mode operation and can only act as initiator during SCSI transactions. The SYM53C875 instruction set implementation is not complete and foremost load/store instructions are missing. Assembly language instructions shown with debug level greater than 1 do not comply with the Symbios Logic assembler format.

Attributes

scsi_bus

Required attribute; **read/write** access; type: `object`. Object must implement the 'scsi-bus' interface.

Command List

Commands

info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
status	print status of the device

Command Descriptions

`<SYM53C875>.info`

Synopsis

`<SYM53C875>.info`

Description

Print detailed information about the configuration of the device.

`<SYM53C875>.pci-header — deprecated`

Synopsis

`<SYM53C875>.pci-header [-v]`

Description

This command is deprecated; use `<SYM53C875>.print-pci-config-reg` instead.

<SYM53C875>.print-pci-config-reg

Synopsis

<SYM53C875>.print-pci-config-reg [-v]

Description

Print the PCI configuration registers. The *-v* flag turns on verbose mode.

<SYM53C875>.status

Synopsis

<SYM53C875>.status

Description

Print detailed information about the current status of the device.

symtable

Provided By
[symtable](#)

Interfaces Implemented

[conf_object](#), [log_object](#), [path_translate](#), [symtable](#)

Description

Symbolic debugging information for an address space.

Command List

Commands

abi	set ABI to use
host-source-at	return the source code position for a given address
list	list source and/or disassemble
load-symbols	load symbols from file
plain-symbols	read raw symbols in nm format
pos	address of line or function
source-path	set source search path for debug info
whereis	find symbol by address

Command Descriptions

<symtable>.abi

Synopsis

`<symtable>.abi ["abi"]`

Description

Set the ABI to *abi*, or show the current ABI. This is normally not needed when loading debug info from a binary, but is needed when using [plain-symbols](#).

See Also

[<symtable>.plain-symbols](#)

<symtable>.host-source-at

Synopsis

`<symtable>.host-source-at address`

Description

Return the source code position [*file*, *line*, *function*] corresponding to a given address.

If the address did not match any file, FALSE will be returned.

If the file was not found, *file* will be FALSE.

<symtable>.list

Synopsis

```
<symtable>.list [-s|-d] [address|"location"] [maxlines]
```

Description

List the source code corresponding to a given address, function or line. The location can be specified as *line* or *file : line* (list from that line); *function* or *file : function* (list that function); or *address* (list from that address).

At most *maxlines* lines of source or asm are printed. *-s* produces source intermixed with disassembly, and *-d* disassembly only.

See Also

[disassemble](#), [whereis](#), [pos](#), [symval](#), [sym-source](#), [load-symbols](#), [sym-value](#)

<symtable>.load-symbols

Synopsis

```
<symtable>.load-symbols file [start] [-t] [-s]
```

Description

Loads symbolic debugging information from *file*. If *start* is given, it is the (absolute) starting address of the file's code. If *-t* is specified, *start* is interpreted as the address of the .text section (similarly to GDB); otherwise *start* is taken to be the address of the first executable segment. If *-s* is given, only symbols are loaded and not any debug information. The Simics search path is used to locate *file*.

See Also

[new-symtable](#), [<symtable>.plain-symbols](#)

<symtable>.plain-symbols

Synopsis

```
<symtable>.plain-symbols file [start]
```

Description

Reads "plain" symbols in BSD **nm(1)** output format. Each line looks like:

value type symbol

where *value* is the symbol value (usually the address) in hex, and *type* is a code letter describing the type of the symbol: **B**, **D** or **R** are treated as data symbols, and **T** as text (code) symbols. File-local symbols have their type letters in lower case, global symbols in upper case. If *start* is specified, it is added to the value of each symbol.

See Also

[new-symtable](#), [<symtable>.load-symbols](#)

<symtable>.pos

Synopsis

<symtable>.pos (*line*|“*function*”)

Description

Finds the address of a source line or a function in a given file (e.g., `myfile.c:4711` or `myfile.c:myfunction`). If only a line number is specified, the command looks at the current value of the program counter to determine the current file. It may be necessary to put the argument inside double quotes for it to be parsed correctly.

See Also

[stack-trace](#), [psym](#), [symval](#)

<symtable>.source-path

Synopsis

<symtable>.source-path [“*path*”]

Description

Sets the source search path to *path*. If no *path* is given, the current search path will be returned. The source search path is used when translating file names in debug information to the appropriate place in the host machine’s file system.

Each component of the (semicolon-separated) search path is either a path name, added to the beginning of source file names, or a string of the form *a>b*, replacing *a* with *b* at the beginning of source file names. Both *a* and *b* will be interpreted as directory name.

Example 1: The source path `/usr/src>/pkg/source;/usr>/misc` would cause the file name `/usr/src/any.c` to be translated to `/pkg/source/any.c`, or, if no such file exists, to `/misc/src/any.c`.

Example 2: The source path `/usr/src>C:\My\Source` would translate the file name `/usr/src/any/file.c` to `C:\My\Source\any\file.c`.

Example 3: The source path `/usr/src` would translate the file name `/tmp/file.c` to `/usr/src/file.c`.

See Also

[<symtable>.load-symbols](#), [new-symtable](#)

<symtable>.whereis

Synopsis

<symtable>.whereis [-d] *address*

Description

Displays the function or variable and (if possible) the source file and line number corresponding to *address*. If `-d` is specified, searches only data symbols. Does not look for stack-allocated variables.

See Also

[pos](#), [psym](#)

sync_domain

Provided By

[Simics Core](#)

Interfaces Implemented

[conf_object](#), [log_object](#)

Description

A controller object for a group of synchronized nodes.

Command List

Commands

[info](#) print information about the device

[status](#) print status of the device

Command Descriptions

<sync_domain>.info

Synopsis

`<sync_domain>.info`

Description

Print detailed information about the configuration of the device.

<sync_domain>.status

Synopsis

`<sync_domain>.status`

Description

Print detailed information about the current status of the device.

system_panel_bool_in

Provided By
[system-panel](#)

Interfaces Implemented
conf_object, [log_object](#)

Description
control panel signal state objects

Attributes

panel
Required attribute; **read/write** access; type: [os] or object. Undocumented

Command List

Commands
[info](#) print information about the device
[status](#) print status of the device

Command Descriptions

`<system_panel_bool_in>.info`

Synopsis
`<system_panel_bool_in>.info`

Description
Print detailed information about the configuration of the device.

`<system_panel_bool_in>.status`

Synopsis
`<system_panel_bool_in>.status`

Description
Print detailed information about the current status of the device.

system_panel_bool_out

Provided By

[system-panel](#)

Interfaces Implemented

`conf_object`, [log_object](#), `signal`

Description

control panel signal state objects

Attributes

panel

Required attribute; **read/write** access; type: [os] or object. Undocumented

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

`<system_panel_bool_out>.info`

Synopsis

`<system_panel_bool_out>.info`

Description

Print detailed information about the configuration of the device.

`<system_panel_bool_out>.status`

Synopsis

`<system_panel_bool_out>.status`

Description

Print detailed information about the current status of the device.

system_panel_number_in

Provided By
[system-panel](#)

Interfaces Implemented
conf_object, [log_object](#)

Description
control panel number state

Attributes

panel
Required attribute; **read/write** access; type: [os] or object. Undocumented

Command List

Commands
[info](#) print information about the device
[status](#) print status of the device

Command Descriptions

`<system_panel_number_in>.info`

Synopsis
`<system_panel_number_in>.info`

Description
Print detailed information about the configuration of the device.

`<system_panel_number_in>.status`

Synopsis
`<system_panel_number_in>.status`

Description
Print detailed information about the current status of the device.

system_panel_number_out

Provided By

[system-panel](#)

Interfaces Implemented

[conf_object](#), [log_object](#), [uint64_state](#)

Description

control panel number state

Attributes

panel

Required attribute; **read/write** access; type: [os] or object. Undocumented

restriction

Optional attribute; **read/write** access; type: integer or nil. If set to a nonnegative integer value, input values greater or equal to this value are forbidden and will cause an error log. If set to nil, any integer is accepted as input. A restriction makes it possible to use the object for widgets that only support a limited range.

Command List

Commands

[info](#) print information about the device

[status](#) print status of the device

Command Descriptions

<system_panel_number_out>.info

Synopsis

<system_panel_number_out>.info

Description

Print detailed information about the configuration of the device.

<system_panel_number_out>.status

Synopsis

<system_panel_number_out>.status

Description

Print detailed information about the current status of the device.

system_panel_state_manager

Provided By
[system-panel](#)

Interfaces Implemented
conf_object, [log_object](#)

Description

Store panel objects whose state have changed, and propagate them to frontend when ready.

Attributes

notification_sent

Pseudo attribute; **read/write** access; type: boolean. This flag indicates if state changes notification has been sent to front end. If true, means state changes existed locally and the notification of the changes has been sent to front end.

panel_component

Required attribute; **read/write** access; type: [os] or object. The containing component. Used to look up slot names for panel input.

use_recorder

Optional attribute; **read/write** access; type: boolean. If this attribute is true, the recorder is used for all input. If this is false, the recorder is bypassed.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<system_panel_state_manager>.info

Synopsis

<system_panel_state_manager>.info

Description

Print detailed information about the configuration of the device.

<system_panel_state_manager>.status

Synopsis

<system_panel_state_manager>.status

Description

Print detailed information about the current status of the device.

tcf-agent

Provided By
[tcf-agent](#)

Interfaces Implemented
conf_object, [log_object](#)

Description
TCF Agent

Attributes

parameters

Required attribute; **read/write** access; type: `string`. Parameter for the TCF agent.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<tcf-agent>.info

Synopsis

`<tcf-agent>.info`

Description

Print detailed information about the configuration of the device.

<tcf-agent>.status

Synopsis

`<tcf-agent>.status`

Description

Print detailed information about the current status of the device.

tcf-context-proxy

Provided By

[tcf-agent](#)

Interfaces Implemented

[conf_object](#), [log_object](#)

Description

TCF Context Proxy

Attributes

agent

Required attribute; **read/write** access; type: `object`. The TCF agent owning the TCF context this object is a proxy for

cid

Required attribute; **read/write** access; type: `string`. The TCF context ID for the context this object is a proxy for

desc

Required attribute; **read/write** access; type: `[so]` or `[soi]`. Description of what this object is a proxy for. Either `["simobj", object]` or `["sw-node", software object, node ID]`.

Command List

Commands

break-line	Add breakpoint at a source code line
break-location	Add breakpoint at a location
info	print information about the device
status	print status of the device

Command Descriptions

`<tcf-context-proxy>.break-line`

Synopsis

`<tcf-context-proxy>.break-line "line"`

Description

Add a breakpoint at a given source code line. The *line* should be given as file-name:linenumber[:columnnumber].

With the `-x`, `-r`, `-w` flags you can specify if the breakpoint should be an execution breakpoint, a read breakpoint, a write breakpoint, or any combination. The default is an execution breakpoint.

The command returns the *id* of the new breakpoint. This can be used to manage and delete the breakpoint using the **breakpoints** object.

See Also

[break-location](#)

<tcf-context-proxy>.break-location

Synopsis

<tcf-context-proxy>.break-location ("location" | address) [length] [-r] [-w] [-x]

Description

Add a breakpoint on a particular location. The location can be specified either as a memory address or a C expression. In addition you can specify the size of the breakpoint, i.e. how many consecutive memory addresses it should match with *size*.

With the *-x*, *-r*, *-w* flags you can specify if the breakpoint should be an execution breakpoint, a read breakpoint, a write breakpoint, or any combination. The default is an execution breakpoint.

The command returns the *id* of the new breakpoint. This can be used to manage and delete the breakpoint using the **breakpoints** object.

See Also

[break-line](#)

<tcf-context-proxy>.info

Synopsis

<tcf-context-proxy>.info

Description

Print detailed information about the configuration of the device.

<tcf-context-proxy>.status

Synopsis

<tcf-context-proxy>.status

Description

Print detailed information about the current status of the device.

telnet_console

Provided By

[telnet-console](#)

Aliases

server-console

Interfaces Implemented

[conf_object](#), [log_object](#), [serial_device](#)

Description

Class allowing telnet access to a simulated serial port.

Attributes

device

Required attribute; **read/write** access; type: `object` or `nil`. Serial device the console is connected to.

new_port_if_busy

Optional attribute; **read/write** access; type: `boolean`. Determines if a new TCP port may be used when restoring from a checkpoint and the saved port is already busy. The default value is true. It may be set to false for setups that rely on the same port being used all the time.

port

Optional attribute; **read/write** access; type: `integer`. TCP port for telnet console server.

recorder

Required attribute; **read/write** access; type: `object`. Recorder device for recording and playback of serial input.

Command List

Commands

break	set a string to break on
capture-start	capture output to file
capture-stop	stop output capture to file
disconnect	terminates the telnet session
info	print information about the device
input	send string to a console
input-file	input a file into a console
kbd-abort	send a keyboard abort signal
list-break-strings	list all active string breakpoints

playback-start	start traffic generator
playback-stop	stop traffic generation
record-start	start recording of output on the console
record-stop	stop recording of output on the console
status	print status of the device
switch-to-host-serial-console	<i>deprecated</i> —
switch-to-serial-link	<i>deprecated</i> —
switch-to-text-console	<i>deprecated</i> —
unbreak	stop breaking on string
unbreak-id	remove a breakpoint
wait-for-string	wait for a string in a script branch
wait-then-write	wait for a string, then write an input string

Command Descriptions

<telnet_console>.break

Synopsis

<telnet_console>.break “*string*” [-once]

Description

Set a string to break on for the console. The simulation will stop when this string is written on the console. If text output is intermixed with control sequences for colors/positioning, the break command may fail breaking on the string.

See Also

<text-console>.unbreak, <text-console>.list-break-strings

<telnet_console>.capture-start

Synopsis

<telnet_console>.capture-start *filename* [-overwrite]

Description

Capture all output from the console to a file.

See Also

<text-console>.capture-stop

<telnet_console>.capture-stop

Synopsis

<telnet_console>.capture-stop

Description

Stop capturing all output from the console.

See Also

[`<text-console>.capture-start`](#)

`<telnet_console>.disconnect`

Synopsis

`<telnet_console>.disconnect`

Description

Terminates the current telnet session and starts waiting for a new telnet client to connect.

`<telnet_console>.info`

Synopsis

`<telnet_console>.info`

Description

Print detailed information about the configuration of the device.

`<telnet_console>.input`

Synopsis

`<telnet_console>.input "string" [delay]`

Description

Send string to a text console. The first character is delayed by 0.01 seconds by default, to wait for the simulated software to be ready after a previous break on string. To input the string immediately, specify delay = 0. The delay is given in seconds. There is no delay inserted between every character.

`<telnet_console>.input-file`

Synopsis

`<telnet_console>.input-file file [-binary]`

Description

Inputs the contents of *file* into the text console. If the *-binary* flag is specified, the file is treated as a binary file.

`<telnet_console>.kbd-abort`

Synopsis

`<telnet_console>.kbd-abort`

Description

Send a serial line abort signal to the attached device. This is only useful when the text-console is connected to a serial device.

<telnet_console>.list-break-strings

Synopsis

<telnet_console>.list-break-strings

Description

Print a list of all active string breakpoints.

See Also

[`<text-console>.break`](#), [`<text-console>.unbreak`](#)

<telnet_console>.playback-start

Synopsis

<telnet_console>.playback-start *filename*

Description

Starts generating bus traffic from a specified file. The file should consist of data and delay specifications in a “command: data” format. Examples: **Delay: 0.345** will delay 345 milliseconds **Hexdata: 72 6f 6f 74 0a** will send the string “root\n” to the connected device. Unrecognised lines are ignored.

See Also

[`<telnet_console>.playback-stop`](#)

<telnet_console>.playback-stop

Synopsis

<telnet_console>.playback-stop

Description

Stop bus traffic generation previously started with playback-start.

See Also

[`<telnet_console>.playback-start`](#)

<telnet_console>.record-start

Synopsis

<telnet_console>.record-start

Description

Start recording of output on the console. All previously recorded output will be discarded. The recorded string can be read with the **record-stop** command.

See Also

[`<text-console>.record-stop`](#)

<telnet_console>.record-stop

Synopsis

<telnet_console>.record-stop

Description

Stop recording of output on the console, and return the recorded string.

See Also

[**<text-console>.record-start**](#)

<telnet_console>.status

Synopsis

<telnet_console>.status

Description

Print detailed information about the current status of the device.

<telnet_console>.switch-to-host-serial-console — *deprecated*

Synopsis

<telnet_console>.switch-to-host-serial-console [*port*]

Description

This command is deprecated; use [**<std-telnet-console>.switch-to-host-serial-console**](#) or [**<std-text-console>.switch-to-host-serial-console**](#) instead.

<telnet_console>.switch-to-serial-link — *deprecated*

Synopsis

<telnet_console>.switch-to-serial-link “*linkname*”

Description

This command is deprecated; use [a manual component connection to a serial link](#) instead.

<telnet_console>.switch-to-text-console — *deprecated*

Synopsis

<telnet_console>.switch-to-text-console

Description

This command is deprecated; use [**<std-telnet-console>.switch-to-text-console**](#) or [**<std-host-serial-console>.switch-to-text-console**](#) instead.

<telnet_console>.unbreak

Synopsis

<telnet_console>.unbreak “*string*”

Description

Stop breaking on specified string.

See Also

[`<text-console>.break`](#), [`<text-console>.list-break-strings`](#)

<telnet_console>.unbreak-id

Synopsis

<telnet_console>.unbreak-id *id*

Description

Remove the breakpoint with specified id.

See Also

[`<text-console>.break`](#), [`<text-console>.list-break-strings`](#)

<telnet_console>.wait-for-string

Synopsis

<telnet_console>.wait-for-string [-always] “*string*”

Description

Wait for the output of the text *string* on the console. This command can only be run from a script branch where it suspends the branch until the string has been found in the output. The *-always* flag is internal and should not be used.

See Also

[`<text-console>.break`](#), [`<text-console>.wait-then-write`](#), [`script-branch`](#), [`wait-for-hap`](#)

<telnet_console>.wait-then-write

Synopsis

<telnet_console>.wait-then-write [-s] “*output-string*” “*input-string*”

Description

Wait for the output of the text *output-string* on the console. When the text is found, write the *input-string*. If the *-s* flag is used, the input string is entered “slowly”, one character at a time, waiting for each character to be echoed back before typing next. This command can only be run from a script branch where it suspends the branch until the string has been found in the output.

See Also

[`<text-console>.wait-for-string`](#), [`<text-console>.break`](#), [`script-branch`](#), [`wait-for-hap`](#)

telnet_frontend

Provided By

[telnet-frontend](#)

Interfaces Implemented

[conf_object](#), [log_object](#)

Description

Class implementing telnet access to the Simics command line. The **telnet_frontend** should only be used in a secure environment since the port is open for telnet access from any user.

Attributes

interactive

Optional attribute; **read/write** access; type: `boolean`. Set to TRUE, default, if future telnet connections will be interactive and FALSE if not. Non-interactive connections are useful for scripted control of the command line since no formatting and color codes are used.

max_connections

Optional attribute; **read/write** access; type: `integer`. The maximum number of allowed telnet connections. Once this number of connections have been created, no further ones are accepted until the *num_connections* attribute is reset. If the value is zero, no limit is set.

num_connections

Session attribute; **read/write** access; type: `integer`. The number of connections done so far since the object was created or the count was reset by a write to this attribute.

port

Optional attribute; **read/write** access; type: `integer`. TCP port for telnet frontend.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

`<telnet_frontend>.info`

Synopsis

`<telnet_frontend>.info`

Description

Print detailed information about the configuration of the device.

```
<telnet_frontend>.status
```

Synopsis

```
<telnet_frontend>.status
```

Description

Print detailed information about the current status of the device.

terminal_frontend

Provided By

[Simics Core](#)

Interfaces Implemented

`conf_object`, [log_object](#)

Description

The **terminal_frontend** class provides access to a Simics command line using a generic VT100/ANSI interface. Input and output to the user is handled by a separate object, specified by the *frontend* attribute, that must implement the `terminal_client` interface and communicate with the **terminal_frontend** object using the `terminal_server` interface.

Attributes

frontend

Required attribute; **read/write** access; type: `object` or `nil`. Object responsible for presenting terminal output to the user and reading input. This object must implement the `terminal_client` interface.

session_id

Required attribute; **read/write** access; type: `integer`. The identifier of the session in the object specified by the *frontend* attribute. Needed since such objects may have multiple concurrent sessions active.

text-console

Provided By
[xterm-console](#)

Aliases
xterm-console

Interfaces Implemented
conf_object, [extended_serial](#), [log_object](#), [serial_device](#)

Description

The **text-console** class provides a text user interface to a simulated machine. The console is typically connected to a simulated serial device. On Windows hosts, a text selection in the console is automatically copied to the clipboard. Text from the clipboard can be inserted in the console by pressing the middle mouse button. On Unix hosts, the console is implemented using an [xterm](#).

Attributes

backspace_is_delete

Optional attribute; **read/write** access; type: boolean. Set if the backspace key (0x08) should send the delete (0x7f) ASCII code. Default is FALSE.

break_string

Pseudo attribute; **write-only** access; type: string. An output string causing Simics to trigger a Xterm_Break_String hap when the string is printed.

char_1c_is_abort

Optional attribute; **read/write** access; type: boolean. Generate a keyboard abort signal when <ctrl>-/ is pressed.

device

Optional attribute; **read/write** access; type: [os], object, or nil. The device connected to the console, must implement one of the 'serial_device' or 'keyboard' interfaces. Only one of the link and device attributes can be set.

link

Optional attribute; **read/write** access; type: object or nil. The serial link object the console is connected to. Only one of the link and device attributes can be set.

no_window

Optional attribute; **read/write** access; type: boolean. If this flag is TRUE, no window will be opened. If it is FALSE, the output is redirected according to the output_file and quiet attributes.

quiet

Optional attribute; **read/write** access; type: boolean. If the no_window attribute is

set, this attribute controls whether or not the console output is redirected to Simics console. If set to TRUE, output is not redirected. If set to FALSE, output is redirected unless the `output_file` is set. If the `no_window` attribute is FALSE, then this attribute has no effect.

recorder

Required attribute; **read/write** access; type: `object`. Recorder device for recording and playback of mouse and keyboard input.

title

Required attribute; **read/write** access; type: `string`. The window title.

Command List

Commands

<code>break</code>	set a string to break on
<code>capture-start</code>	capture output to file
<code>capture-stop</code>	stop output capture to file
<code>close</code>	close console window
<code>disable-quiet</code>	enable output redirection
<code>disable-read-only</code>	disable read-only mode
<code>enable-quiet</code>	disable output redirection
<code>enable-read-only</code>	enable read-only mode
<code>info</code>	print information about the device
<code>input</code>	send string to a console
<code>input-file</code>	input a file into a console
<code>kbd-abort</code>	send a keyboard abort signal
<code>list-break-strings</code>	list all active string breakpoints
<code>open</code>	open console window
<code>playback-start</code>	start traffic generator
<code>playback-stop</code>	stop traffic generation
<code>quiet</code>	<i>deprecated</i> — toggle console output redirection
<code>read-only</code>	<i>deprecated</i> — toggle read-only mode
<code>record-start</code>	start recording of output on the console
<code>record-stop</code>	stop recording of output on the console
<code>status</code>	print status of the device
<code>switch-to-host-serial-console</code>	<i>deprecated</i> —
<code>switch-to-serial-link</code>	<i>deprecated</i> —
<code>switch-to-telnet-console</code>	<i>deprecated</i> —
<code>unbreak</code>	stop breaking on string
<code>unbreak-id</code>	remove a breakpoint
<code>wait-for-string</code>	wait for a string in a script branch
<code>wait-then-write</code>	wait for a string, then write an input string

Command Descriptions

`<text-console>.break`

Synopsis

`<text-console>.break "string" [-once]`

Description

Set a string to break on for the console. The simulation will stop when this string is written on the console. If text output is intermixed with control sequences for colors/positioning, the break command may fail breaking on the string.

See Also

[`<text-console>.unbreak`](#), [`<text-console>.list-break-strings`](#)

`<text-console>.capture-start`

Synopsis

`<text-console>.capture-start filename [-overwrite]`

Description

Capture all output from the console to a file.

See Also

[`<text-console>.capture-stop`](#)

`<text-console>.capture-stop`

Synopsis

`<text-console>.capture-stop`

Description

Stop capturing all output from the console.

See Also

[`<text-console>.capture-start`](#)

`<text-console>.close`

Synopsis

`<text-console>.close`

Description

Close the console window. When the console window is closed, any output is redirected to the main Simics window. This can be suppressed with the `<text-console>.quiet` command. To provide input to the target machine while the console is closed, use the `<text-console>.input` command.

See Also

[`<text-console>.quiet`](#), [`<text-console>.open`](#)

`<text-console>.disable-quiet`

Synopsis

`<text-console>.disable-quiet`
`<text-console>.enable-quiet`

Description

When a **text-console** window is closed, all output on that target console is by default redirected to the Simics console. This redirection can be turned off by enabling quiet mode with the `<text-console>.enable-quiet` command and turned on again using `<text-console>.disable-quiet`.

See Also

[`<text-console>.close`](#), [`<text-console>.capture-start`](#)

`<text-console>.disable-read-only`

Synopsis

`<text-console>.disable-read-only`
`<text-console>.enable-read-only`

Description

Change the read-only mode of the console. Input is only accepted if the console is in read-write mode. In read-only mode, all input is discarded, except input generated by the `<text-console>.insert` command.

See Also

[`<text-console>.input`](#)

`<text-console>.enable-quiet`

Synopsis

`<text-console>.enable-quiet`
`<text-console>.disable-quiet`

Description

When a **text-console** window is closed, all output on that target console is by default redirected to the Simics console. This redirection can be turned off by enabling quiet mode with the `<text-console>.enable-quiet` command and turned on again using `<text-console>.disable-quiet`.

See Also

[`<text-console>.close`](#), [`<text-console>.capture-start`](#)

<text-console>.enable-read-only

Synopsis

```
<text-console>.enable-read-only  
<text-console>.disable-read-only
```

Description

Change the read-only mode of the console. Input is only accepted if the console is in read-write mode. In read-only mode, all input is discarded, except input generated by the <text-console>.insert command.

See Also

[`<text-console>.input`](#)

<text-console>.info

Synopsis

```
<text-console>.info
```

Description

Print detailed information about the configuration of the device.

<text-console>.input

Synopsis

```
<text-console>.input "string" [delay]
```

Description

Send string to a text console. The first character is delayed by 0.01 seconds by default, to wait for the simulated software to be ready after a previous break on string. To input the string immediately, specify delay = 0. The delay is given in seconds. There is no delay inserted between every character.

<text-console>.input-file

Synopsis

```
<text-console>.input-file file [-binary]
```

Description

Inputs the contents of *file* into the text console. If the *-binary* flag is specified, the file is treated as a binary file.

<text-console>.kbd-abort

Synopsis

```
<text-console>.kbd-abort
```

Description

Send a serial line abort signal to the attached device. This is only useful when the text-console is connected to a serial device.

`<text-console>.list-break-strings`

Synopsis

`<text-console>.list-break-strings`

Description

Print a list of all active string breakpoints.

See Also

[`<text-console>.break`](#), [`<text-console>.unbreak`](#)

`<text-console>.open`

Synopsis

`<text-console>.open`

Description

Open the console window

See Also

[`<text-console>.quiet`](#), [`<text-console>.close`](#)

`<text-console>.playback-start`

Synopsis

`<text-console>.playback-start` *filename*

Description

Starts generating bus traffic from a specified file. The file should consist of data and delay specifications in a “command: data” format. Examples: **Delay: 0.345** will delay 345 milliseconds **Hexdata: 72 6f 6f 74 0a** will send the string “root\n” to the connected device. Unrecognised lines are ignored.

See Also

[`<text-console>.playback-stop`](#)

`<text-console>.playback-stop`

Synopsis

`<text-console>.playback-stop`

Description

Stop bus traffic generation previously started with playback-start.

See Also

[`<text-console>.playback-start`](#)

`<text-console>.quiet — deprecated`

Synopsis

[`<text-console>.quiet`](#)

Description

This command is deprecated; use [`<text-console>.enable-quiet`](#) or [`<text-console>.disable-quiet`](#) instead.

`<text-console>.read-only — deprecated`

Synopsis

[`<text-console>.read-only`](#)

Description

This command is deprecated; use [`<text-console>.enable-read-only`](#) or [`<text-console>.disable-read-only`](#) instead.

See Also

[`<text-console>.input`](#)

`<text-console>.record-start`

Synopsis

[`<text-console>.record-start`](#)

Description

Start recording of output on the console. All previously recorded output will be discarded. The recorded string can be read with the **record-stop** command.

See Also

[`<text-console>.record-stop`](#)

`<text-console>.record-stop`

Synopsis

[`<text-console>.record-stop`](#)

Description

Stop recording of output on the console, and return the recorded string.

See Also

[`<text-console>.record-start`](#)

```
<text-console>.status
```

Synopsis

```
<text-console>.status
```

Description

Print detailed information about the current status of the device.

```
<text-console>.switch-to-host-serial-console — deprecated
```

Synopsis

```
<text-console>.switch-to-host-serial-console [port]
```

Description

This command is deprecated; use [`<std-telnet-console>.switch-to-host-serial-console`](#) or [`<std-text-console>.switch-to-host-serial-console`](#) instead.

```
<text-console>.switch-to-serial-link — deprecated
```

Synopsis

```
<text-console>.switch-to-serial-link “linkname”
```

Description

This command is deprecated; use [a manual component connection](#) to a serial link instead.

```
<text-console>.switch-to-telnet-console — deprecated
```

Synopsis

```
<text-console>.switch-to-telnet-console [port]
```

Description

This command is deprecated; use [`<std-host-serial-console>.switch-to-telnet-console`](#) or [`<std-text-console>.switch-to-telnet-console`](#) instead.

```
<text-console>.unbreak
```

Synopsis

```
<text-console>.unbreak “string”
```

Description

Stop breaking on specified string.

See Also

[`<text-console>.break`](#), [`<text-console>.list-break-strings`](#)

<text-console>.unbreak-id

Synopsis

<text-console>.unbreak-id *id*

Description

Remove the breakpoint with specified id.

See Also

[`<text-console>.break`](#), [`<text-console>.list-break-strings`](#)

<text-console>.wait-for-string

Synopsis

<text-console>.wait-for-string [-always] “*string*”

Description

Wait for the output of the text *string* on the console. This command can only be run from a script branch where it suspends the branch until the string has been found in the output. The *-always* flag is internal and should not be used.

See Also

[`<text-console>.break`](#), [`<text-console>.wait-then-write`](#), [`script-branch`](#), [`wait-for-hap`](#)

<text-console>.wait-then-write

Synopsis

<text-console>.wait-then-write [-s] “*output-string*” “*input-string*”

Description

Wait for the output of the text *output-string* on the console. When the text is found, write the *input-string*. If the *-s* flag is used, the input string is entered “slowly”, one character at a time, waiting for each character to be echoed back before typing next. This command can only be run from a script branch where it suspends the branch until the string has been found in the output.

See Also

[`<text-console>.wait-for-string`](#), [`<text-console>.break`](#), [`script-branch`](#), [`wait-for-hap`](#)

text_panel_frontend

Provided By

[system-panel-text](#)

Interfaces Implemented

`conf_object`, [log_object](#)

Description

A system panel frontend that works in text mode.

Command List

Commands

[info](#) print information about the device

[status](#) print status of the device

Command Descriptions

`<text_panel_frontend>.info`

Synopsis

`<text_panel_frontend>.info`

Description

Print detailed information about the configuration of the device.

`<text_panel_frontend>.status`

Synopsis

`<text_panel_frontend>.status`

Description

Print detailed information about the current status of the device.

time-server

Provided By

[time-server-c](#)

Interfaces Implemented

`conf_object`, [log_object](#)

Description

The time-server class provides virtual time services to programs running outside of Simics. It provides those services through a text protocol on a user defined TCP port.

Attributes

port

Required attribute; **read/write** access; type: `integer` or `nil`. TCP port for server. Use 0 for any port (the actual port can be read back from this attribute). Set to Nil to disable the server.

trace-mem-hier

Provided By

[trace](#)

Interfaces Implemented

[conf_object](#), [log_object](#), [snoop_memory](#), [timing_model](#)

Description

This class is defined in the trace module. It is used by the tracer to listen to memory traffic on each CPU.

trace-sync

Provided By
[trace-sync](#)

Interfaces Implemented
conf_object, [log_object](#)

Description

Provides a workaround for a hardware bug present in Intel® Core™2 processors and later.

Attributes

cpu

Required attribute; **read/write** access; type: object. CPU being synchronized.

recorder

Required attribute; **read/write** access; type: object. Recorder object used to store time synchronizations.

Command List

Commands

disable	disable core2 hardware bug workaround
enable	enable core2 hardware bug workaround
info	print information about the device
status	print status of the device

Command Descriptions

<trace-sync>.disable

Synopsis

```
<trace-sync>.disable  
<trace-sync>.enable
```

Description

Enables trace synchronization.

This module provides a workaround for a hardware determinism problem present in certain Core2 Duo processors.

<trace-sync>.enable

Synopsis

```
<trace-sync>.enable  
<trace-sync>.disable
```

Description

Enables trace synchronization.

This module provides a workaround for a hardware determinism problem present in certain Core2 Duo processors.

<trace-sync>.info

Synopsis

<trace-sync>.info

Description

Print detailed information about the configuration of the device.

<trace-sync>.status

Synopsis

<trace-sync>.status

Description

Print detailed information about the current status of the device.

trans-sorter

Provided By
[g-cache](#)

Interfaces Implemented
conf_object, [log_object](#), [timing_model](#)

Description

A trans-sorter object divides the transactions in 4 transaction flows according to two criteria: are they cacheable or not? Are they generated by the cpu or by a device? They are forwarded to the four timing models attributes corresponding to the four possibilities. This can be useful to ignore some types of transactions when tracing or connecting a memory hierarchy.

Attributes

cache

Required attribute; **read/write** access; type: object or nil. Cache to which the sorter is connected.

trans-splitter

Provided By
[g-cache](#)

Interfaces Implemented
conf_object, [log_object](#), [timing_model](#)

Description

A trans-splitter object should be inserted between two caches if the higher-level cache has a larger cache line size than the lower-level cache, or between the processor and the id-splitter object if you have a processor that can do unaligned accesses or accesses larger than one cache line. It splits cacheable transactions that span more than one naturally aligned next-cache-line-size bytes big lines into multiple transactions that each fit within one line.

Attributes

cache

Required attribute; **read/write** access; type: `object` or `nil`. Cache to which the splitter is connected.

next_cache_line_size

Required attribute; **read/write** access; type: `integer`. Cache line size used for splitting transactions.

timing_model

Required attribute; **read/write** access; type: `object` or `nil`. Object listening on transactions coming from the splitter.

trans-staller

Provided By
[trans-staller](#)

Interfaces Implemented
conf_object, [log_object](#), [timing_model](#)

Description

This class implements a simple transaction staller. If a transaction is stallable, it will return a fixed stall time. It can be used for a simple memory simulation at the end of a cache hierarchy.

TSB12LV26

Provided By
[TSB12LV26](#)

Interfaces Implemented

`conf_object`, `firewire_device`, `io_memory`, `log_object`

Ports

PHY ([int_register](#), [io_memory](#)), csr_aliases ([int_register](#), [io_memory](#)), firewire_config_registers ([int_register](#), [io_memory](#)), ohci ([int_register](#), [io_memory](#)), pci_config ([int_register](#), [io_memory](#)), ti_extensions ([int_register](#), [io_memory](#))

Description

The TSB12V26 1394 (FireWire) OHCI Host Controller

Attributes

config_registers

Pseudo attribute; **read-only** access; type: `[i*]`. The PCI configuration registers, each 32 bits in size.

expansion_rom_size

Optional attribute; **read/write** access; type: `integer`. The size of the expansion ROM mapping.

pci_bus

Required attribute; **read/write** access; type: `[os]` or `object`. The PCI bus this device is connected to, implementing the `pci-bus` interface.

pci_config_command

Optional attribute; **read/write** access; type: `integer`. The PCI command register.

pci_config_device_id

Optional attribute; **read/write** access; type: `integer`. The Device ID of the PCI device

pci_config_vendor_id

Optional attribute; **read/write** access; type: `integer`. The Vendor ID of the PCI device

Command List

Commands

info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
status	print status of the device

Command Descriptions

<TSB12LV26>.info

Synopsis

<TSB12LV26>.info

Description

Print detailed information about the configuration of the device.

<TSB12LV26>.pci-header — *deprecated*

Synopsis

<TSB12LV26>.pci-header [-v]

Description

This command is deprecated; use [**<TSB12LV26>.print-pci-config-reg**](#)s instead.

<TSB12LV26>.print-pci-config-reg

Synopsis

<TSB12LV26>.print-pci-config-reg [-v]

Description

Print the PCI configuration registers. The *-v* flag turns on verbose mode.

<TSB12LV26>.status

Synopsis

<TSB12LV26>.status

Description

Print detailed information about the current status of the device.

tsi500

Provided By

[tsi500](#)

Interfaces Implemented

[conf_object](#), [log_object](#)

Ports

Port0 ([rapidio_v3](#)), Port1 ([rapidio_v3](#)), Port2 ([rapidio_v3](#)), Port3 ([rapidio_v3](#)), regs ([int_register](#))

Description

Tundra Tsi500 4-port RapidIO switch. Supports 8-bit addressing.

Attributes

devices

Optional attribute; **read/write** access; type: [o|[os]|n{4}]. Receivers on the bus. Must implement the rapidio_v3 interface.

Command List

Commands

[info](#) print information about the device

[status](#) print status of the device

Command Descriptions

<tsi500>.info

Synopsis

[<tsi500>.info](#)

Description

Print detailed information about the configuration of the device.

<tsi500>.status

Synopsis

[<tsi500>.status](#)

Description

Print detailed information about the current status of the device.

usb_disk

Provided By
[usb-disk](#)

Interfaces Implemented
conf_object, [log_object](#)

Description

A standard USB SCSI disk.

Attributes

bytes_transferred

Optional attribute; **read/write** access; type: `integer`. Number of bytes sent in data stage.

command_status

Optional attribute; **read/write** access; type: `integer`. The execution status of the command block.

current_cbw

Optional attribute; **read/write** access; type: `data` or `nil`. The current handled CBW.

descriptor_data

Optional attribute; **read/write** access; type: `data` or `nil`. The descriptors in data format.

device_address

Optional attribute; **read/write** access; type: `integer`. USB device/function address.

device_qualifier_descriptor_data

Optional attribute; **read/write** access; type: `data` or `nil`. The device qualifier in data format.

device_state

Optional attribute; **read/write** access; type: `integer`. USB device state.

hid_report_descriptor_data

Optional attribute; **read/write** access; type: `data` or `nil`. The descriptors in data format.

scsi_data

Optional attribute; **read/write** access; type: `data` or `nil`. Buffered SCSI data.

scsi_disk

Required attribute; **read/write** access; type: `[os]` or `object`. Undocumented

string_descriptor_array

Optional attribute; **read/write** access; type: [d*]. String descriptors.

transport_state

Optional attribute; **read/write** access; type: integer. State for mass storage command/data/status protocol.

usb_host

Optional attribute; **read/write** access; type: [os], object, or nil. The USB host controller.

usb_hs_keyboard

Provided By

[usb-input-devices](#)

Interfaces Implemented

conf_object, [keyboard](#), [log_object](#)

Description

A High-Speed USB keyboard device.

Attributes

descriptor_data

Optional attribute; **read/write** access; type: data or nil. The descriptors in data format.

device_address

Optional attribute; **read/write** access; type: integer. USB device/function address.

device_qualifier_descriptor_data

Optional attribute; **read/write** access; type: data or nil. The device qualifier in data format.

device_state

Optional attribute; **read/write** access; type: integer. USB device state.

hid_data

Optional attribute; **read/write** access; type: [iiiiiiii]. Keyboard HID data.

hid_report_descriptor_data

Optional attribute; **read/write** access; type: data or nil. The descriptors in data format.

key_queue

Optional attribute; **read/write** access; type: [i*]. contains key events which can not be reported yet without loosing previous key events

string_descriptor_array

Optional attribute; **read/write** access; type: [d*]. String descriptors.

tr_table

Optional attribute; **read/write** access; type: [i*]. Keyboard translation table

usb_host

Optional attribute; **read/write** access; type: [os], object, or nil. The USB host controller.

Command List

Commands

info	print information about the device
key-down	
key-press	
key-up	
status	print status of the device

Command Descriptions

`<usb_hs_keyboard>.info`

Synopsis

`<usb_hs_keyboard>.info`

Description

Print detailed information about the configuration of the device.

`<usb_hs_keyboard>.key-down`

Synopsis

`<usb_hs_keyboard>.key-down key-code`
`<usb_hs_keyboard>.key-up key-code`
`<usb_keyboard>.key-up key-code`

Description

Send a key press to the keyboard controller. The argument is the internal Simics keycode. The `<sun-keyboard>.key-press` command is recommend instead.

See Also

[`<usb_hs_keyboard>.key-press`](#)

`<usb_hs_keyboard>.key-press`

Synopsis

`<usb_hs_keyboard>.key-press "key" ... [simultaneously]`

Description

Press one or more keys on the keyboard. This translates to a series of key down events followed by the matching key up events. The names correspond to the keys on a U.S. keyboard with no keys remapped by software. If simultaneously is true, that means you press keys simultaneously, otherwise, you press keys sequentially. Default of simultaneously is false.

`<usb_hs_keyboard>.key-up`

Synopsis

```
<usb_hs_keyboard>.key-up key-code  
<usb_hs_keyboard>.key-down key-code  
<usb_keyboard>.key-up key-code
```

Description

Send a key press to the keyboard controller. The argument is the internal Simics keycode. The `<sun-keyboard>.key-press` command is recommend instead.

See Also

[`<usb_hs_keyboard>.key-press`](#)

`<usb_hs_keyboard>.status`

Synopsis

```
<usb_hs_keyboard>.status
```

Description

Print detailed information about the current status of the device.

usb_keyboard

Provided By

[usb-input-devices](#)

Interfaces Implemented

conf_object, [keyboard](#), [log_object](#)

Description

A USB keyboard device.

Attributes

descriptor_data

Optional attribute; **read/write** access; type: data or nil. The descriptors in data format.

device_address

Optional attribute; **read/write** access; type: integer. USB device/function address.

device_qualifier_descriptor_data

Optional attribute; **read/write** access; type: data or nil. The device qualifier in data format.

device_state

Optional attribute; **read/write** access; type: integer. USB device state.

hid_data

Optional attribute; **read/write** access; type: [iiiiiiii]. Keyboard HID data.

hid_report_descriptor_data

Optional attribute; **read/write** access; type: data or nil. The descriptors in data format.

key_queue

Optional attribute; **read/write** access; type: [i*]. contains key events which can not be reported yet without loosing previous key events

string_descriptor_array

Optional attribute; **read/write** access; type: [d*]. String descriptors.

tr_table

Optional attribute; **read/write** access; type: [i*]. Keyboard translation table

usb_host

Optional attribute; **read/write** access; type: [os], object, or nil. The USB host controller.

Command List

Commands

info	print information about the device
key-down	
key-press	
key-up	
status	print status of the device

Command Descriptions

`<usb_keyboard>.info`

Synopsis

`<usb_keyboard>.info`

Description

Print detailed information about the configuration of the device.

`<usb_keyboard>.key-down`

Synopsis

`<usb_keyboard>.key-down key-code`

Description

Send a key press to the keyboard controller. The argument is the internal Simics keycode. The `<sun-keyboard>.key-press` command is recommend instead.

See Also

[`<usb_hs_keyboard>.key-press`](#)

`<usb_keyboard>.key-press`

Synopsis

`<usb_keyboard>.key-press "key" ... [simultaneously]`

Description

Press one or more keys on the keyboard. This translates to a series of key down events followed by the matching key up events. The names correspond to the keys on a U.S. keyboard with no keys remapped by software. If simultaneously is true, that means you press keys simultaneously, otherwise, you press keys sequentially. Default of simultaneously is false.

`<usb_keyboard>.key-up`

Synopsis

```
<usb_keyboard>.key-up key-code  
<usb_hs_keyboard>.key-down key-code  
<usb_hs_keyboard>.key-up key-code
```

Description

Send a key press to the keyboard controller. The argument is the internal Simics keycode. The `<sun-keyboard>.key-press` command is recommend instead.

See Also

[`<usb_hs_keyboard>.key-press`](#)

`<usb_keyboard>.status`

Synopsis

```
<usb_keyboard>.status
```

Description

Print detailed information about the current status of the device.

usb_mouse

Provided By
[usb-input-devices](#)

Interfaces Implemented
[abs_pointer](#), [conf_object](#), [log_object](#)

Description
A USB mouse device.

Attributes

descriptor_data
Optional attribute; **read/write** access; type: `data` or `nil`. The descriptors in data format.

device_address
Optional attribute; **read/write** access; type: `integer`. USB device/function address.

device_qualifier_descriptor_data
Optional attribute; **read/write** access; type: `data` or `nil`. The device qualifier in data format.

device_state
Optional attribute; **read/write** access; type: `integer`. USB device state.

hid_data
Optional attribute; **read/write** access; type: `[iii]`. Mouse HID data.

hid_report_descriptor_data
Optional attribute; **read/write** access; type: `data` or `nil`. The descriptors in data format.

string_descriptor_array
Optional attribute; **read/write** access; type: `[d*]`. String descriptors.

usb_host
Optional attribute; **read/write** access; type: `[os]`, `object`, or `nil`. The USB host controller.

Command List

Commands
[info](#) print information about the device
[status](#) print status of the device

Command Descriptions

<usb_mouse>.info

Synopsis

<usb_mouse>.info

Description

Print detailed information about the configuration of the device.

<usb_mouse>.status

Synopsis

<usb_mouse>.status

Description

Print detailed information about the current status of the device.

usb_santa

Provided By
[usb-santa](#)

Interfaces Implemented
conf_object, [log_object](#)

Description

An USB device in the form of Santa Claus that glows in the night.

Attributes

descriptor_data

Optional attribute; **read/write** access; type: data or nil. The descriptors in data format.

device_address

Optional attribute; **read/write** access; type: integer. USB device/function address.

device_qualifier_descriptor_data

Optional attribute; **read/write** access; type: data or nil. The device qualifier in data format.

device_state

Optional attribute; **read/write** access; type: integer. USB device state.

hid_report_descriptor_data

Optional attribute; **read/write** access; type: data or nil. The descriptors in data format.

string_descriptor_array

Optional attribute; **read/write** access; type: [d*]. String descriptors.

usb_host

Optional attribute; **read/write** access; type: [os], object, or nil. The USB host controller.

usb_tablet

Provided By
[usb-tablet](#)

Interfaces Implemented
[abs_pointer](#), [conf_object](#), [log_object](#)

Description
A USB tablet device.

Attributes

descriptor_data

Optional attribute; **read/write** access; type: `data` or `nil`. The descriptors in data format.

device_address

Optional attribute; **read/write** access; type: `integer`. USB device/function address.

device_qualifier_descriptor_data

Optional attribute; **read/write** access; type: `data` or `nil`. The device qualifier in data format.

device_state

Optional attribute; **read/write** access; type: `integer`. USB device state.

hid_report_descriptor_data

Optional attribute; **read/write** access; type: `data` or `nil`. The descriptors in data format.

string_descriptor_array

Optional attribute; **read/write** access; type: `[d*]`. String descriptors.

tablet_hid_data

Optional attribute; **read/write** access; type: `[iiiiii]`. Tablet HID data.

usb_host

Optional attribute; **read/write** access; type: `[os]`, `object`, or `nil`. The USB host controller.

Command List

Commands

info print information about the device
status print status of the device

Command Descriptions

<usb_tablet>.info

Synopsis

<usb_tablet>.info

Description

Print detailed information about the configuration of the device.

<usb_tablet>.status

Synopsis

<usb_tablet>.status

Description

Print detailed information about the current status of the device.

usb_wacom_tablet

Provided By

[usb-wacom-tablet](#)

Interfaces Implemented

[abs_pointer](#), [conf_object](#), [log_object](#)

Description

A USB Wacom tablet device.

Attributes

descriptor_data

Optional attribute; **read/write** access; type: `data` or `nil`. The descriptors in data format.

device_address

Optional attribute; **read/write** access; type: `integer`. USB device/function address.

device_qualifier_descriptor_data

Optional attribute; **read/write** access; type: `data` or `nil`. The device qualifier in data format.

device_state

Optional attribute; **read/write** access; type: `integer`. USB device state.

hid_report_descriptor_data

Optional attribute; **read/write** access; type: `data` or `nil`. The descriptors in data format.

string_descriptor_array

Optional attribute; **read/write** access; type: `[d*]`. String descriptors.

tablet_hid_data

Optional attribute; **read/write** access; type: `[iiiiiiii]`. Tablet HID data.

usb_host

Optional attribute; **read/write** access; type: `[os]`, `object`, or `nil`. The USB host controller.

Command List

Commands

[info](#) print information about the device

[status](#) print status of the device

Command Descriptions

<usb_wacom_tablet>.info

Synopsis

<usb_wacom_tablet>.info

Description

Print detailed information about the configuration of the device.

<usb_wacom_tablet>.status

Synopsis

<usb_wacom_tablet>.status

Description

Print detailed information about the current status of the device.

vga_pci

Provided By
[vga-pci](#)

Interfaces Implemented

`conf_object`, [io_memory](#), [log_object](#)

Description

The VGA-PCI device is identical to the VGA device except it connects to a PCII bus instead of the ISA bus. This functionality of this device can also be inherited by more advanced graphic devices such as the accel-vga device. Limitations: Not all possible video modes are tested. The CGA Compatibility modes are not yet supported. Different character maps are not yet supported. Read Mode 1 is not yet implemented. Panning is not supported. Spilt Screen is not supported. Blinking text is not supported. Blinking color in video mode 0xf is not supported. Underlining in MDA compatibility mode is not supported. Extend ninth bit in text mode is not implemented.

Attributes

console

Required attribute; **read/write** access; type: `object` or `nil`. Console object that must implement either `gfx_console` or both the `serial_device` and `extended_serial` interfaces.

image

Required attribute; **read/write** access; type: `object`. Image object containing the VRAM.

memory_space

Required attribute; **read/write** access; type: `object`. Memory space to which the device is mapped. Needed for dynamic remapping.

Command List

Commands

info	print information about the device
pci-header	<i>deprecated</i> — print PCI device header
print-pci-config-reg	print PCI configuration registers
redraw	Redraw display
refresh-rate	Set rate at which vga_pci updated display
status	print status of the device
text-dump	Print text contents of display
wait-for-string	Wait for substring in text mode

Command Descriptions

`<vga_pci>.info`

Synopsis

`<vga_pci>.info`

Description

Print detailed information about the configuration of the device.

`<vga_pci>.pci-header — deprecated`

Synopsis

`<vga_pci>.pci-header [-v]`

Description

This command is deprecated; use `<vga_pci>.print-pci-config-reg`s instead.

`<vga_pci>.print-pci-config-reg`s

Synopsis

`<vga_pci>.print-pci-config-reg [-v]`

Description

Print the PCI configuration registers. The `-v` flag turns on verbose mode.

`<vga_pci>.redraw`

Synopsis

`<vga_pci>.redraw`

Description

This command sends the current frame buffer contents of the simulated video device to the graphics console. If a simulated cursor is active, it is updated as well.

`<vga_pci>.refresh-rate`

Synopsis

`<vga_pci>.refresh-rate [rate]`

Description

Set the rate at which the `vga_pci` device updates the display.

The default is 1000 Hz (simulated). NOTE: The rate is currently the same for all `vga_pci` devices.

```
<vga_pci>.status
```

Synopsis

```
<vga_pci>.status
```

Description

Print detailed information about the current status of the device.

```
<vga_pci>.text-dump
```

Synopsis

```
<vga_pci>.text-dump
```

Description

Print the contents of the display in text mode.

```
<vga_pci>.wait-for-string
```

Synopsis

```
<vga_pci>.wait-for-string [-always] "substring" [sample_interval]
```

Description

Samples text mode output every *sample_interval* virtual seconds and returns if *substring* is present in the output. The *-always* flag is internal and should not be used.

vmcom

Provided By

[vmcom](#)

Interfaces Implemented

[abs_pointer](#), [conf_object](#), [io_memory](#), [log_object](#)

Description

Implements the VMware tools protocol. Currently only supports the mouse protocol.

Attributes

current_mouse_state

Optional attribute; **read/write** access; type: [iiii]. Current mouse state.

last_mouse_state

Optional attribute; **read/write** access; type: [iiii]. Last mouse state that was enqueued for delivery to software.

mouse

Required attribute; **read/write** access; type: [os] or object. Object that implements the abs-pointer-activate interface.

mouse_data_queue

Optional attribute; **read/write** access; type: [i*]. Mouse data enqueued for delivery to software.

mouse_enabled

Optional attribute; **read/write** access; type: boolean. Keeps track of whether the mouse protocol is enabled.

recorder

Required attribute; **read/write** access; type: [os] or object. Recorder device for playback of input.

Command List

Commands

[info](#) print information about the device
[status](#) print status of the device

Command Descriptions

`<vmcom>.info`

Synopsis

`<vmcom>.info`

Description

Print detailed information about the configuration of the device.

<vmcom>.status**Synopsis****<vmcom>.status****Description**

Print detailed information about the current status of the device.

wdb-remote

Provided By
[wdb-remote](#)

Interfaces Implemented
conf_object, [log_object](#)

Description

Debug agent that implements the Wind River Debug protocol. When started, it acts as a WDB agent, which makes it possible to control Simics from a WDB client, e.g. Wind River's Tornado or Workbench.

Attributes

bootline

Required attribute; **read/write** access; type: string. Boot line string

cpu_type

Required attribute; **read/write** access; type: string. CPU type

host_pool_base

Required attribute; **read/write** access; type: integer. Host pool base

host_pool_size

Required attribute; **read/write** access; type: integer. Host pool size

int_size

Required attribute; **read/write** access; type: string. Int size on target, in bytes

long_size

Required attribute; **read/write** access; type: string. Long size on target, in bytes

mem_size

Required attribute; **read/write** access; type: integer. Memory size

pointer_size

Required attribute; **read/write** access; type: string. Pointer size on target, in bytes

rt_name

Required attribute; **read/write** access; type: string. Runtime name

rt_type

Required attribute; **read/write** access; type: string. Runtime type

rt_version

Required attribute; **read/write** access; type: string. Runtime version string

Command List

Commands

- info** print information about the device
- status** print status of the device

Command Descriptions

<wdb-remote>.info

Synopsis

<wdb-remote>.info

Description

Print detailed information about the configuration of the device.

<wdb-remote>.status

Synopsis

<wdb-remote>.status

Description

Print detailed information about the current status of the device.

Chapter 6

Interfaces

This chapter lists the interface types defined by this package. Note that the C definitions are shown here, including the first `conf_object_t *` argument to the methods. That argument is automatically added when calling interface methods from DML or Python, and must be omitted.

Interfaces are defined using the `SIM_INTERFACE (name)` macro. The macro will define a type called `name_interface_t`, which shall be used as the type name. The macro expands to an equivalent of the following:

```
typedef struct name_interface {
    // methods go here
} name_interface_t;
```

abs_pointer

Implemented By

[usb_mouse](#), [usb_tablet](#), [usb_wacom_tablet](#), [vmcom](#)

Description

Interface implemented by tablet devices. Used by consoles to send touchpad events to the controller. The *set_state* function is called when something changes in the console. The coordinates are given as scaled absolute scaled values, where (0, 0) is the upper-left corner and (0xffff, 0xffff) is the lower-right corner.

```
typedef enum {
    ABS_POINTER_BUTTON_LEFT    = 0x20,
    ABS_POINTER_BUTTON_RIGHT   = 0x10,
    ABS_POINTER_BUTTON_MIDDLE  = 0x08
} abs_pointer_buttons_t;

typedef struct {
    abs_pointer_buttons_t buttons;
    uint16 x;
    uint16 y;
    uint16 z;
} abs_pointer_state_t;

SIM_INTERFACE(abs_pointer) {
    void (*set_state)(conf_object_t *obj, abs_pointer_state_t state);
};

#define ABS_POINTER_INTERFACE "abs_pointer"
```

Execution Context

Instruction Context for all methods.

abs_pointer_activate

Implemented By

[gfx-console](#)

Description

Interface used by tablet controllers to temporary turn off and on the tracking of absolute pointer locations. Implemented by consoles. When disabled, no calls will be made to the controller's abs_pointer interface.

```
SIM_INTERFACE(abs_pointer_activate) {  
    void (*enable)(conf_object_t *obj);  
    void (*disable)(conf_object_t *obj);  
};  
  
#define ABS_POINTER_ACTIVATE_INTERFACE "abs_pointer_activate"
```

Execution Context

Instruction Context for all methods.

address_profiler

Implemented By

[branch_recorder](#), [data-profiler](#)

Description

Interface for getting statistics out of profilers. The target is some kind of profiler whose data can be meaningfully viewed as counts per address.

The function *num_views* returns the number k of different ways you can view the data of this object. The view selection parameter *view* to all other functions in the interface accepts values between 0 and $k - 1$.

description returns a short string that explains what the data means. *physical_addresses* returns true if the profiler works with physical addresses, or false if it uses virtual addresses. *address_bits* returns the number of bits in an address.

granularity_log2 returns the base 2 logarithm of the size, in bytes, of the address intervals that the counters are associated to. For example, if the data is instruction execution count and each instruction is 4 bytes long, one would expect the granularity to be at least 4 bytes since that is the smallest interval containing a whole instruction (but it might be more, if the profiler is less fine-grained for some reason). And for a 4-byte granularity, *granularity_log2* would return 2.

sum returns the sum of all counters between *start* and *stop*, inclusive. *max* returns the maximum value of any counter in the range.

iter returns an address profile iterator that will visit all nonzero counters in the range precisely once, in some order. In C, you can use the functions *SIM_iter_next*, *SIM_iter_addr* and *SIM_iter_free* to operate the iterator. In Python, it works just like any other iterator, and returns (count, address) pairs. Note that you may not continue to use the iterator after the underlying address profiler has been modified.

```
SIM_INTERFACE(address_profiler) {
    addr_prof_iter_t *(*iter)(conf_object_t *prof_obj, unsigned view,
                           generic_address_t start,
                           generic_address_t stop);
    uint64 (*sum)(conf_object_t *prof_obj, unsigned view,
                  generic_address_t start, generic_address_t end);
    uint64 (*max)(conf_object_t *prof_obj, unsigned view,
                  generic_address_t start, generic_address_t end);
    unsigned (*granularity_log2)(conf_object_t *prof_obj, unsigned view);
    int (*address_bits)(conf_object_t *prof_obj, unsigned view);
    int (*physical_addresses)(conf_object_t *prof_obj, unsigned view);
    const char *(*description)(conf_object_t *prof_obj, unsigned view);
    unsigned (*num_views)(conf_object_t *prof_obj);
};
```

```
#define ADDRESS_PROFILER_INTERFACE "address_profiler"
```

Execution Context

Instruction Context for all methods.

Command List

address-profile-data	linear map of address profiling data
address-profile-info	general info about an address profiler
address-profile-toplist	print toplist of address profiling data

Command Descriptions

<address_profiler>.address-profile-data

Synopsis

```
<address_profiler>.address-profile-data [view] [address] [cell-bits] [row-bits] [table-bits]
[start] [stop] [lines] [columns] [-same-prefix]
```

Description

Display a map of (a part of) the address space covered by the address profiler, and the counts of one of its views associated with each address. The view is specified by the *view* argument; default is view 0. The default behavior is to display the smallest interval that contains all counts; you can change this with either the *start* and *stop* or the *address* and *cell-bits*, *row-bits* or *table-bits* arguments.

Cells that have zero counts are marked with “.”. Cells that have a non-zero count, but were rounded to zero, are marked with “o”.

If one of *cell-bits*, *row-bits* or *table-bits* is specified, then each cell, or each table row, or the entire table is limited to that many bits of address space. By default the display starts at address 0, but if an address is specified with the *address* argument, the displayed interval is shifted to make that address is visible.

If *start* and *stop* are specified, the display is limited to the smallest interval containing both addresses.

The maximum number of lines and columns in the table are limited by the *lines* and *columns* arguments (the default is 20 lines, 8 columns). The scale of the map is adjusted to fit this limit.

Normally, the display chooses an appropriate prefix for the count of each cell; with the *-same-prefix* flag, all counts will be forced to have the same prefix. This is useful if a lot of small but non-zero values makes it hard to spot the really big values.

<address_profiler>.address-profile-info

Synopsis

```
<address_profiler>.address-profile-info [-sum] [-max]
```

Description

Print general info about an object implementing the address-profiler interface, such as a list of the available views. If the `-sum` or `-max` flags are given, will also print the sum or max of each view over the entire address space.

```
<address_profiler>.address-profile-toplist
```

Synopsis

```
<address_profiler>.address-profile-toplist [symtable] [samples] [start] [stop] [view] [count_interval]
```

Description

Print address profiling regions sorted by count. The `symtable` attribute can be used to map data profiling regions in form of physical addresses to source function and file information.

The `samples` argument specifies the number of sampling points used to create the list containing the highest count. The sampling range is determined by `start` and `stop`. The default values are 100 samples in the interval 0x0–0xffffffffc. The granularity is defined by the data profiler object.

The `view` attribute selects the address profiler view.

The `count_interval` attribute defines the range in which sampled data regions will match even though the data profiler count is not equal. For example, assume that the samples in the region 0x20c–0x20c has a count of 4711 and 4713 in 0x20d–0x20f. These regions will be considered to be one region if `count_interval` is 4, but not if it is 1.

attribute_monitor

Implemented By
[frontend-server](#)

Description

This is a collection of API calls that allows a customized connection between frontend extensions and custom target modules.

The function *register_monitored_attribute* should be called by an object that wishes to communicate the value of an attribute to a frontend plug-in. The *attr_obj* and *attr_name* parameters indicate the attribute to communicate. The function returns a handle that will be used to refer to this instance of the attribute. The function should be called before the object is configured, typically from the *finalize_instance* method.

After an object has registered an attribute with *register_monitored_attribute*, the object should call the function *monitored_attribute_changed* with the returned attribute ID whenever the value of that attribute changes, including from the attribute setter.

```
typedef struct monitored_attribute attribute_id_t;

SIM_INTERFACE(attribute_monitor) {
    attribute_id_t *(*register_monitored_attribute) (
        conf_object_t *obj,
        conf_object_t *attr_obj,
        const char *attr_name);
    void (*monitored_attribute_changed) (
        conf_object_t *obj,
        attribute_id_t *id);
};

#define ATTRIBUTE_MONITOR_INTERFACE "attribute_monitor"
```

Execution Context

<i>register_monitored_attribute</i>	Outside Execution Context
<i>monitored_attribute_changed</i>	Instruction Context

branch_arc

Implemented By
[branch_recorder](#)

Description

Interface for getting branch arcs out profilers. The target is some kind of profiler whose data can be meaningfully viewed as branch arcs (usually a branch profiler).

iter returns a branch arc iterator that will visit all branch arcs in the range precisely once, in order of selected address (to or from, selected with *dir*), other address and type. In Python, it works just like any other iterator, and returns (from, to, counter, type) tuples. Note that you may not continue to use the iterator after the underlying profiler has been modified.

`branch_arc_type_t` defines the branch types returned by a branch arc iterator.

Branch_Arc_Branch

Normal branch operation

Branch_Arc_Exception

Branch because an exception was encountered

Branch_Arc_Exception_Return

Branch to finish an exception handler

```
typedef enum {
    Branch_Arc_Branch,
    Branch_Arc_Exception,
    Branch_Arc_Exception_Return,
    Branch_Arc_Max
} branch_arc_type_t;

typedef enum {
    BR_Direction_From,
    BR_Direction_To
} branch_recorder_direction_t;

SIM_INTERFACE(branch_arc) {
    branch_arc_iter_t *(*iter)(conf_object_t *prof_obj,
                               generic_address_t start,
                               generic_address_t stop,
                               branch_recorder_direction_t dir);
};

#define BRANCH_ARC_INTERFACE "branch_arc"
```

Execution Context

Instruction Context for all methods.

breakpoint

Implemented By

[context](#), [fake-space](#), [memory-space](#)

Description

The breakpoint interface is implemented by any object that supports breaking on an address range.

```
SIM_INTERFACE(breakpoint) {
    void (*insert_breakpoint)(conf_object_t *object,
                             conf_object_t *caller,
                             breakpoint_handle_t handle,
                             access_t access,
                             generic_address_t start,
                             generic_address_t end);
    void (*remove_breakpoint)(conf_object_t *object,
                             breakpoint_handle_t handle);
    breakpoint_info_t (*get_breakpoint)(conf_object_t *obj,
                                       breakpoint_handle_t handle);
};

#define BREAKPOINT_INTERFACE "breakpoint"
```

The **insert_breakpoint** function is called when adding a breakpoint on *object*. The *handle* identified the breakpoint and is to be used when triggering the breakpoint and when breakpoints are removed. The *access* parameter specifies the types of accesses that the breakpoint is set for. The breakpoint range is from *start* to *end* and includes both ends.

The implementer of this interface should call *caller* through the **breakpoint_trigger** interface to trigger the breakpoint.

remove_breakpoint should remove the breakpoint and further accesses to the address range should not trigger that breakpoint.

This interface is only to be used by the Simics core. Other uses of breakpoints should go through the available breakpoint API calls such as **SIM_breakpoint**.

Execution Context

Execution Context for all methods.

Command List

[break](#) set breakpoint

[tbreak](#) set temporary breakpoint on current processor

Command Descriptions

`<breakpoint>.break`

Synopsis

```
<breakpoint>.break address [length] [-r] [-w] [-x]
<breakpoint>.tbreak address [length] [-r] [-w] [-x]
break address [length] [-r] [-w] [-x]
```

Description

Add breakpoint (read, write, or execute) on an object implementing the breakpoint interface. This is typically a memory space object such as physical memory; e.g., `phys_mem0.break 0xff3800`. Accesses intersecting the given range will trigger the breakpoint. By default the breakpoint will only trigger for instruction execution, but any subset of read, write, and execute accesses can be set to trigger using combinations of `-r`, `-w`, and `-x`.

`length` is the interval length in bytes (default is 1).

Breakpoints inserted with the `tbreak` command are automatically disabled when they have triggered.

The default action at a triggered breakpoint is to return to the frontend. This can be changed by using haps. When an execution breakpoint is triggered, Simics will return to the command prompt before the instructions is executed, while instructions triggering read or write breakpoints will complete before control is returned to the command prompt.

To break on a virtual address, use a context object:

`cpu0_context.break 0x1ff00`

Several breakpoints can be set on the same address and Simics will break on them in turn. If hap handlers (callback functions) are connected to the breakpoints they will also be executed in turn. Hap handlers are called before the access is performed, allowing the user to read a memory value that may be overwritten by the access. See the Simics Reference Manual for a description of hap handlers.

Each breakpoint is associated with an id (printed when the breakpoint is set or by the `list-breakpoints` command) which is used for further references to the breakpoint.

For convenience there are also a `break` command which sets a breakpoint on memory connected to the current frontend processor (see `pselect`). Default is to break on virtual address accesses (in the current context). By prefixing the address with `p`: it is possible to break on physical accesses as well (cf. `phys_mem0.break`); e.g., `break p:0xffffc0`.

Several attributes can be set for a breakpoint for breaking only when some conditions are true. See the `disable`, `enable`, `ignore`, `set-prefix`, `set-substr` and `set-pattern` commands for more details.

Breakpoints can be removed using `delete`.

See Also

[unbreak](#), [delete](#), [enable](#), [ignore](#), [set-prefix](#), [set-substr](#), [set-pattern](#), [list-breakpoints](#), [wait-for-breakpoint](#)

<breakpoint>.tbreak

Synopsis

```
<breakpoint>.tbreak address [length] [-r] [-w] [-x]
<breakpoint>.break address [length] [-r] [-w] [-x]
break address [length] [-r] [-w] [-x]
```

Description

Add breakpoint (read, write, or execute) on an object implementing the breakpoint interface. This is typically a memory space object such as physical memory; e.g., **phys_mem0.break 0xff3800**. Accesses intersecting the given range will trigger the breakpoint. By default the breakpoint will only trigger for instruction execution, but any subset of read, write, and execute accesses can be set to trigger using combinations of **-r**, **-w**, and **-x**.

length is the interval length in bytes (default is 1).

Breakpoints inserted with the **tbreak** command are automatically disabled when they have triggered.

The default action at a triggered breakpoint is to return to the frontend. This can be changed by using haps. When an execution breakpoint is triggered, Simics will return to the command prompt before the instructions is executed, while instructions triggering read or write breakpoints will complete before control is returned to the command prompt.

To break on a virtual address, use a context object:

```
cpu0_context.break 0x1ff00
```

Several breakpoints can be set on the same address and Simics will break on them in turn. If hap handlers (callback functions) are connected to the breakpoints they will also be executed in turn. Hap handlers are called before the access is performed, allowing the user to read a memory value that may be overwritten by the access. See the Simics Reference Manual for a description of hap handlers.

Each breakpoint is associated with an id (printed when the breakpoint is set or by the **list-breakpoints** command) which is used for further references to the breakpoint.

For convenience there are also a **break** command which sets a breakpoint on memory connected to the current frontend processor (see **pselect**). Default is to break on virtual address accesses (in the current context). By prefixing the address with **p**: it is possible to break on physical accesses as well (cf. **phys_mem0.break**); e.g., **break p:0xffffc0**.

Several attributes can be set for a breakpoint for breaking only when some conditions are true. See the **disable**, **enable**, **ignore**, **set-prefix**, **set-substr** and **set-pattern** commands for more details.

Breakpoints can be removed using **delete**.

See Also

[unbreak](#), [delete](#), [enable](#), [ignore](#), [set-prefix](#), [set-substr](#), [set-pattern](#), [list-breakpoints](#), [wait-for-breakpoint](#)

bridge

Implemented By

[generic_pcie_switch_port](#), [i21150](#), [i21152](#), [i21154](#), [i21554-prim](#), [i21554-scnd](#), [i21555-prim](#), [i21555-scnd](#)

Description

The `bridge` interface is implemented by objects that bridge between memory spaces. The `not_taken` function is called if the access is not claimed by any device in the destination memory-space. If a memory transaction reaches a mapping that has the same bridge object as the previous mapping, the access is considered as not taken, and the `not_taken` function for the first bridge mapping is called.

```
SIM_INTERFACE(bridge) {
    exception_type_t (*not_taken)(conf_object_t *NOTNULL obj,
                                conf_object_t *NOTNULL src_space,
                                conf_object_t *NOTNULL dst_space,
                                exception_type_t ex,
                                generic_transaction_t *NOTNULL mem_op,
                                map_info_t mapinfo);
};

#define BRIDGE_INTERFACE "bridge"
```

Execution Context

Instruction Context for all methods.

checkpoint

Implemented By
[gfx-console](#), [image](#)

Description

The *save* function in this interface is called when a checkpoint is saved, right before the attributes of an object is read. If defined, it should prepare the object for checkpointing, saving any state to *path* that is not directly included in the attributes. Errors are signalled through exceptions.

The *finish* function is called after the checkpoint has been saved. If *success* is nonzero, the checkpoint was saved successfully; otherwise there was a failure. This permits the object to clean up temporary data structures and files in either case.

The function *has_persistent_data*, if implemented, should return 0 if the object only has volatile attributes, 1 otherwise. This overrides `Sim_Attr_Persistent` on individual attributes.

```
SIM_INTERFACE(checkpoint) {
    void (*save)(conf_object_t *obj, const char *NOTNULL path);
    void (*finish)(conf_object_t *obj, int success);
    int (*has_persistent_data)(conf_object_t *obj);
};

#define CHECKPOINT_INTERFACE "checkpoint"
```

Execution Context

Outside Execution Context for all methods.

component

Implemented By

[cell-and-clocks](#), [component](#), [cp3_quad100tx](#), [datagram_link](#), [ddr-memory-module](#), [ddr2-memory-module](#), [ddr3-memory-module](#), [dummy-component](#), [etg_comp](#), [etg_panel](#), [eth_injector_comp](#), [ethernet_cable](#), [ethernet_hub](#), [ethernet_switch](#), [ethernet_vlan_switch](#), [example-keypad](#), [example-status-panel](#), [generic_PCIE_switch](#), [i2c_link_v2](#), [instruction-data-splitter](#), [isa-fourport](#), [isa-lance](#), [isa-vga](#), [memory-timer](#), [micron_mtfc2ggqdi_emmc_card](#), [micron_mtfc4ggqdi_emmc_card](#), [micron_mtfc4ggqdi_sdhc_card](#), [os_awareness](#), [pc-dual-serial-ports](#), [pc-floppy-controller](#), [pc-quad-serial-ports](#), [pc-single-parallel-port](#), [pci-accel-vga](#), [pci-am79c973](#), [pci-bcm5703c](#), [pci-bcm5704c](#), [pci-dec21041](#), [pci-dec21140a](#), [pci-dec21140a-dml](#), [pci-dec21143](#), [pci-i21152](#), [pci-i82543gc](#), [pci-i82546bg](#), [pci-i82559](#), [pci-ispl040](#), [pci-ispl2200](#), [pci-pd6729](#), [pci-pmc1553](#), [pci-sil680a](#), [pci-sym53c810](#), [pci-sym53c875](#), [pci-sym53c876](#), [pci-tsb12lv26](#), [pci-vga](#), [phy-mii-transceiver](#), [phy_comp](#), [ps2-keyboard-mouse](#), [rapidio_link](#), [real-network-bridge](#), [real-network-host](#), [real-network-router](#), [sample-gcache](#), [sdram-memory-module](#), [ser_link](#), [signal_link](#), [simple-fc-disk](#), [simple_memory_module](#), [sio-w83627hf](#), [std-etg](#), [std-ethernet-link](#), [std-firewire-bus](#), [std-firewire-sample-device](#), [std-generic-link](#), [std-generic-link-sample](#), [std-graphics-console](#), [std-host-serial-console](#), [std-ide-cdrom](#), [std-ide-disk](#), [std-ms1553-link](#), [std-pcmcia-flash-disk](#), [std-scsi-bus](#), [std-scsi-cdrom](#), [std-scsi-disk](#), [std-serial-link](#), [std-server-console](#), [std-service-node](#), [std-super-io](#), [std-telnet-console](#), [std-text-console](#), [std-text-graphics-console](#), [std mmc_card](#), [std_sata_cdrom](#), [std_sata_disk](#), [top-component](#), [usb-disk](#), [usb-santa](#), [usb-tablet-comp](#), [usb-wacom-tablet-comp](#), [usb_hs_keyboard_comp](#), [usb_keyboard_comp](#), [usb_mouse_comp](#), [usb_xmas_tree](#)

Description

All component classes must implement the `component` interface. All functions in the interface must be implemented.

The `pre_instantiate` function is called before the component is instantiated. The function returns `true` if the component can be instantiated, or `false` if not.

The component might need to do some extra work after the component has been instantiated. This should be done when called via the `post_instantiate` function.

The `create_cell` function returns `true` if the configuration system can create a default cell object for the component, or `false` if not. Both `pre_instantiate` and `create_cell` typically return `true`.

Component has slots. A slot has key and value. The key is the slot name as a string. The value is a conf object, a pre conf object, or None, or nested lists of such types.

Slots are either defined in the component or added after the component has been created. Slots defined in the component are static slots which can not be deleted, but the slot value can be changed. Slots added to the component after creation are dynamic slots and they can be removed when wanted.

The `get_slots` function returns a dictionary with slot names as dictionary keys and slot values as dictionary values.

The *get_slot_objects* function returns a list of all conf objects and pre conf objects extracted from all slot values.

The *get_slot_value* returns the slot value. The slot name is passed as *slot* argument. A slot value is set using the *set_slot_value* function. The *value* argument should be a conf object, pre conf object, or None, or nested lists of such types. The get and set functions can generate exceptions on failure.

The *has_slot* function returns `true` if the *slot* exists, otherwise `false`. The slot can either be a static slot or a dynamic slot. The *add_slot* function adds the slot named *slot*. Adding a slot can fail if the slot already exist. The added slot will be a dynamic slot. A dynamic slot can be deleted. The *del_slot* function deletes a dynamic slot. Deleting a slot will fail if the slot does not exist or if the slot is static. Both *add_slot* and *del_slot* returns `true` on success or `false` on failure.

```
SIM_INTERFACE(component) {
    bool (*pre_instantiate)(conf_object_t *obj);
    void (*post_instantiate)(conf_object_t *obj);
    bool (*create_cell)(conf_object_t *obj);

    attr_value_t (*get_slots)(conf_object_t *obj);
    attr_value_t (*get_slot_objects)(conf_object_t *obj);

    attr_value_t (*get_slot_value)(conf_object_t *obj,
                                  const char *NOTNULL slot);
    void (*set_slot_value)(conf_object_t *obj,
                          const char *NOTNULL slot,
                          attr_value_t value);

    bool (*has_slot)(conf_object_t *obj,
                     const char *NOTNULL slot);
    bool (*add_slot)(conf_object_t *obj,
                     const char *NOTNULL slot);
    bool (*del_slot)(conf_object_t *obj,
                     const char *NOTNULL slot);
};

#define COMPONENT_INTERFACE "component"
```

Execution Context

<i>post_instante</i>	Outside Execution Context
<i>pre_instantiate</i>	Outside Execution Context
<i>create_cell</i>	Outside Execution Context
<i>get_slots</i>	Instruction Context
<i>get_objects</i>	Instruction Context
<i>get_slot_value</i>	Outside Execution Context

<i>set_slot_value</i>	Outside Execution Context
<i>add_slot</i>	Outside Execution Context
<i>del_slot</i>	Outside Execution Context

component_connector

Implemented By

[cell-and-clocks](#), [component](#), [cp3_quad100tx](#), [datagram_link](#), [ddr-memory-module](#), [ddr2-memory-module](#), [ddr3-memory-module](#), [dummy-component](#), [etg_comp](#), [eth_injector_comp](#), [ethernet_cable](#), [ethernet_hub](#), [ethernet_switch](#), [ethernet_vlan_switch](#), [example-keypad](#), [example-status-panel](#), [generic_pcie_switch](#), [i2c_link_v2](#), [instruction-data-splitter](#), [isa-fourport](#), [isa-lance](#), [isa-vga](#), [memory-timer](#), [micron_mtfc2ggqdi_emmc_card](#), [micron_mtfc4ggqdi_emmc_card](#), [micron_mtfc4ggqdi_sdhc_card](#), [pc-dual-serial-ports](#), [pc-floppy-controller](#), [pc-quad-serial-ports](#), [pc-single-parallel-port](#), [pci-accel-vga](#), [pci-am79c973](#), [pci-bcm5703c](#), [pci-bcm5704c](#), [pci-dec21041](#), [pci-dec21140a](#), [pci-dec21140a-dml](#), [pci-dec21143](#), [pci-i21152](#), [pci-i82543gc](#), [pci-i82546bg](#), [pci-i82559](#), [pci-ispl040](#), [pci-ispl200](#), [pci-pd6729](#), [pci-pmc1553](#), [pci-sil680a](#), [pci-sym53c810](#), [pci-sym53c875](#), [pci-sym53c876](#), [pci-tsbl2lv26](#), [pci-vga](#), [phy-mii-transceiver](#), [phy_comp](#), [ps2-keyboard-mouse](#), [rapidio_link](#), [real-network-bridge](#), [real-network-host](#), [real-network-router](#), [sample-gcache](#), [sdram-memory-module](#), [ser_link](#), [signal_link](#), [simple-fc-disk](#), [simple-memory-module](#), [sio-w83627hf](#), [std-etg](#), [std-etherent-link](#), [std-firewire-bus](#), [std-firewire-sample-device](#), [std-generic-link](#), [std-generic-link-sample](#), [std-graphics-console](#), [std-host-serial-console](#), [std-ide-cdrom](#), [std-ide-disk](#), [std-ms1553-link](#), [std-pcmcia-flash-disk](#), [std-scsi-bus](#), [std-scsi-cdrom](#), [std-scsi-disk](#), [std-serial-link](#), [std-server-console](#), [std-service-node](#), [std-super-io](#), [std-telnet-console](#), [std-text-console](#), [std-text-graphics-console](#), [std mmc_card](#), [std_sata_cdrom](#), [std_sata_disk](#), [top-component](#), [usb-disk](#), [usb-santa](#), [usb-tablet-comp](#), [usb-wacom-tablet-comp](#), [usb_hs_keyboard_comp](#), [usb_keyboard_comp](#), [usb_mouse_comp](#), [usb_xmas_tree](#)

Description

The `component_connector` is implemented by components that use connector objects for handling connections between components.

The connection setup is made in two stages, the check stage and the connect stage. The check stage is often not needed, but it can be used to make sure that the later connect step will not fail. Each connection is handled by a connector object. The connector object will both handle the connection in both direction, i.e. sending connect information and receiving connector information. Two components that should be connected must implement one connector object each.

The `get_check_data` and `get_connect_data` will be called from the connector object to get connection data to send to the other part of the connection, i.e. to the destination. The data sent must be an `attr_value_t` type.

The `check`, `connect`, and `disconnect` functions are called from the connector object when another connector wants to connect to this connection. The connection data is passed as the `attr` argument.

```
SIM_INTERFACE(component_connector) {
    attr_value_t (*get_check_data)(conf_object_t *obj,
                                  conf_object_t *NOTNULL connector);
    attr_value_t (*get_connect_data)(conf_object_t *obj,
```

```
                                conf_object_t *NOTNULL connector);
bool (*check) (conf_object_t *obj, conf_object_t *NOTNULL connector,
               attr_value_t attr);
void (*connect) (conf_object_t *obj, conf_object_t *NOTNULL connector,
                  attr_value_t attr);
void (*disconnect) (conf_object_t *obj,
                    conf_object_t *NOTNULL connector);
};

#define COMPONENT_CONNECTOR_INTERFACE "component_connector"
```

Execution Context

Outside Execution Context for all methods.

connector

Implemented By

[connector](#), [dynamic_link_connector](#), [static_link_connector](#)

Description

The `connector` interface must be implemented by connector objects. The interface describes the connector and how it can be connected to other connectors. A connector is used for connecting component objects. The connector is just a proxy for the connection, the connector use the `component_connector` interface in the components to setup the connection.

The `type` function returns the connector type as a string. Two connectors can only be connected if the type is identical.

A hotpluggable connector returns `true` when calling the `hotpluggable` function, otherwise `false`. A hotpluggable connector can be connected before or after the component is instantiated, a non hotpluggable connector must be connected before the component is instantiated.

The `required` function returns `true` if the connector must be connected before instantiated, otherwise `false`.

A connector can be connected to many connectors, but it is only supported if the `multi` function return `true`. It is not recommended that a connector support multi connections. A component can instead dynamically create connectors when needed.

A connector has a direction; `up`, `down` or `any`. The direction decides in which order the connectors in a component tree are connected and the structure of the tree.

Connections are setup in two steps. The first step is to add the connection, which is done using the `add_destination` function. Adding a connection can fail for several reasons and it is implementation dependent how the connection can fail. A connection can for instance fail because the destination object requires something that the source component did not pass when checking the connection. Just adding a connection does not mean that the components actually connect. The components has to save the data given with the `add_destination` function. The actual setup of the connection is made in the second step when the `update` function is called, which finalizes the connection.

The `add_destination` and `remove_destination` functions sets the state of the connection. It is first when the `update` function is called when the connection is finalized. Both `add_destination` and `remove_destination` returns `true` if the call was successful otherwise they return `false`.

Parameters to the `add_destination` function are the own object `obj` and the destination object `dst_obj`. The destination object must be a port object and it must implement the `connector` interface.

The `update` function is called when the component should update its current connection status. The status of the connection has previously been set using the `add_destination` or/and `remove_destination` functions.

The *destination* function returns a list of port objects that the its connected to. The *destination* function returns the state of the connection not the finalized state, i.e. the state of the connection before *update* functional call.

The *check*, *connect*, and *disconnect* functions initiates a connection setup via the connector object. The connector will forward the setup to the components affected by the connection via the `component_connector` interface.

The *deletion_requested* function is called after disconnecting components. A dynamic connector might want to return True in order to be deleted just after the disconnection.

```

typedef enum {
    Sim_Connector_Direction_Up,
    Sim_Connector_Direction_Down,
    Sim_Connector_Direction_Any
} connector_direction_t;

SIM_INTERFACE(connector) {
    char *(*type)(conf_object_t *obj);
    bool (*hotpluggable)(conf_object_t *obj);
    bool (*required)(conf_object_t *obj);
    bool (*multi)(conf_object_t *obj);
    connector_direction_t (*direction)(conf_object_t *obj);

    bool (*add_destination)(conf_object_t *obj, conf_object_t *connector);
    bool (*remove_destination)(conf_object_t *obj,
                               conf_object_t *connector);
    attr_value_t (*destination)(conf_object_t *obj); /* list of
                                                       connector objects */
    void (*update)(conf_object_t *obj);

    bool (*check)(conf_object_t *obj, attr_value_t attr);
    void (*connect)(conf_object_t *obj, attr_value_t attr);
    void (*disconnect)(conf_object_t *obj);
    bool (*deletion_requested)(conf_object_t *obj);
};

#define CONNECTOR_INTERFACE "connector"

```

Execution Context

Outside Execution Context for all methods.

cycle

Implemented By
[clock](#)

Description

Interface Methods

The `cycle` interface is typically implemented by processors, but can be implemented by other objects as well. Its purpose is to handle events based on time. The cycle queue has a cycle as the smallest time unit. The cycle queue also has an associated frequency which makes it possible to define events based on seconds or picoseconds.

The `get_frequency()` function returns the frequency in Hertz for the `queue`. Most objects implementing `cycle` also have a notification mechanism for frequency changes through the `simple_dispatcher` interface in the `cpu_frequency` port. It is recommended that such a notification mechanism is used to get updates rather than polling with `get_frequency()`.

The current number of cycles executed by the `queue` is returned by `get_cycle_count()`. Time elapsed since the queue was created is returned by `get_time()` (in seconds) and `get_time_in_ps` (in picoseconds); this will be equal to `get_cycle_count()` divided by `get_frequency()` if the frequency has been constant since time zero.

The `cycles_delta()` function returns the highest number of cycles `obj` can run *before* it passes the absolute local time `when`, assuming no frequency change occurs in the meantime. Note that `cycles_delta()` can raise an exception if `when` was too far ahead in the future. The `cycles_delta_from_ps()` function performs the same function, for an absolute local time expressed in picoseconds.

The `post_cycle()` function will schedule an event that will occur after `cycles` counted from local current time at `queue`. The `post_time()` is similar but takes `seconds` as argument, while `post_time_in_ps()` takes a number of `picoseconds`.

An event previously posted can be removed by calling `cancel()`. The `cancel()` function takes a function `pred` as argument which is called when a matching event is found. The event is only removed if `pred` returns 1.

The `find_next_cycle()`, `find_next_time()` and `find_next_time_as_ps()` functions take the same arguments as `cancel()` but only returns the number of cycles, seconds or picoseconds before the event occur. The `evclass` is the event class, `obj` is the object posting the event, and `user_data` is pointer to data used as a parameter when calling the callback function defined in the `evclass`.

The `events()` method returns a list of all pending events in expiration order. Each element is a four-element list containing the event object, the event class name, the expiration time counted in cycles as an integer and the event description as given by the event class `describe()` method, or NIL for events whose event class do not define that method.

What happens to already posted events when a frequency change occurs is implementation dependent. Simics processors will scale the cycle queue to keep the

time left before triggering events equal across the frequency change. Note that the new times will be rounded to entire cycles during the scaling, so approximations may occur when switching between high and low frequencies.

Implementation

It is implementation dependent how the queue is implemented, whether cycles or seconds are used as underlying time unit, and what happens when the frequency is changed.

Objects implementing the cycle interface are usually meant to be scheduled by Simics itself. For this to happen, a number of conditions must be fulfilled:

- Each schedulable object implementing the `cycle` interface must be controlled by an object implementing the `execute` interface. It can be the same object that implements the `execute` interface. The object implementing the `execute` interface points to the object implementing the `cycle` interface via its `queue` attribute.
- Any schedulable object implementing the `cycle` interface must inform Simics about changes in frequency by calling the `VT_clock_frequency_change()` function. That also applies to the initial frequency set when the object is created.
- For schedulable objects, the `cycle` interface must be registered with `SIM_register_clock()`, which will also add some Simics specific attributes to the corresponding class. Beyond those, the implementor of the `cycle` can use any checkpoint representation. The `name` field in the event class data structure is unique, and the attribute setter function for checkpoint restore can use `VT_get_event_class()` to get the event class structure corresponding to an event class name.

```
SIM_INTERFACE(cycle) {
    cycles_t (*get_cycle_count)(conf_object_t *queue);
    double (*get_time)(conf_object_t *queue);
    cycles_t (*cycles_delta)(conf_object_t *NOTNULL clock,
                             double when);

    uint64 (*get_frequency)(conf_object_t *queue);

    void (*post_cycle)(
        conf_object_t *NOTNULL queue,
        event_class_t *NOTNULL evclass,
        conf_object_t *NOTNULL obj,
        cycles_t cycles,
        lang_void *user_data);
    void (*post_time)(
        conf_object_t *NOTNULL queue,
```

```

        event_class_t *NOTNULL evclass,
        conf_object_t *NOTNULL obj,
        double seconds,
        lang_void *user_data);

void (*cancel) (
    conf_object_t *NOTNULL queue,
    event_class_t *NOTNULL evclass,
    conf_object_t *NOTNULL obj,
    int (*pred)(lang_void *data, lang_void *match_data),
    lang_void *match_data);

cycles_t (*find_next_cycle) (
    conf_object_t *NOTNULL queue,
    event_class_t *NOTNULL evclass,
    conf_object_t *NOTNULL obj,
    int (*pred)(lang_void *data, lang_void *match_data),
    lang_void *match_data);

double (*find_next_time) (
    conf_object_t *NOTNULL queue,
    event_class_t *NOTNULL evclass,
    conf_object_t *NOTNULL obj,
    int (*pred)(lang_void *data, lang_void *match_data),
    lang_void *match_data);

attr_value_t (*events)(conf_object_t *NOTNULL obj);

/* new picoseconds based functions */
local_time_t (*get_time_in_ps)(conf_object_t *queue);
cycles_t (*cycles_delta_from_ps)(conf_object_t *NOTNULL clock,
                                 local_time_t when);
void (*post_time_in_ps) (
    conf_object_t *NOTNULL queue,
    event_class_t *NOTNULL evclass,
    conf_object_t *NOTNULL obj,
    duration_t picoseconds,
    lang_void *user_data);

duration_t (*find_next_time_in_ps) (
    conf_object_t *NOTNULL queue,
    event_class_t *NOTNULL evclass,
    conf_object_t *NOTNULL obj,
    int (*pred)(lang_void *data, lang_void *match_data),
    lang_void *match_data);

```

```
};  
  
#define CYCLE_INTERFACE "cycle"
```

Execution Context

Instruction Context for all methods.

Command List

cycle-break	set cycle breakpoint
cycle-break-absolute	set absolute cycle breakpoint
print-time	print number of steps and cycles executed
wait-for-cycle	wait until reaching cycle
wait-for-time	wait until reaching a specified time

Command Descriptions

<cycle>.cycle-break

Alias

`<cycle>.cb`

Synopsis

`<cycle>.cycle-break cycles`
`cycle-break [cpu-name] cycles`

Description

Sets a breakpoint so that the CPU will stop after running *cycles* number of cycles from the time the command was issued. If the CPU is not specified the selected frontend processor will be used (see [pselect](#)).

To list all breakpoints set use the command [list-breakpoints](#).

See Also

[cycle-break-absolute](#), [step-break](#), [step-break-absolute](#), [list-breakpoints](#)

<cycle>.cycle-break-absolute

Alias

`<cycle>.cba`

Synopsis

`<cycle>.cycle-break-absolute cycles`
`cycle-break-absolute [cpu-name] cycles`

Description

Set a breakpoint so that the selected CPU will stop after its cycle counter has reached the *cycles* value. If the CPU is not specified the selected frontend processor will be used (see [pselect](#)).

To list all breakpoints set use the command [list-breakpoints](#).

See Also

[cycle-break](#), [step-break](#), [step-break-absolute](#), [list-breakpoints](#)

<cycle>.print-time

Alias

<cycle>.ptime

Synopsis

<cycle>.print-time [-s] [-c] [-t]

Description

Prints the number of steps and cycles that a processor has executed. The cycle count is also displayed as simulated time in seconds.

If called from a processor namespace (e.g., **cpu0.print-time**), the time for that processor is printed. Otherwise, the time for the current processor is printed, or, if the **-all** flag is given, the time for all processors.

If the **-c** flag used, the cycle count for the processor is returned and nothing is printed. The **-s** flag is similar and returns the step count and **-t** returns the time in seconds as a floating point value.

A step is a completed instruction or an exception. An instruction that fails to complete because of an exception will count as a single step, including the exception.

<cycle>.wait-for-cycle

Synopsis

<cycle>.wait-for-cycle [-always] *cycle* [-relative]

Description

Postpones execution of a script branch until the processor reaches the specified cycle in the simulation. If *relative* is specified, the branch will be suspended until the processor has executed the specified number of cycles instead. The **-always** flag is internal and should not be used.

See Also

[script-branch](#), [<step>.wait-for-step](#)

<cycle>.wait-for-time

Synopsis

<cycle>.wait-for-time [-always] *seconds* [-relative]

Description

Postpones execution of a script branch until the processor reaches the specified time, in seconds, from the start of the simulation. If *relative* is given, the branch is suspended for a specified duration instead. The **-always** flag is internal and should not be used.

See Also

[script-branch](#), [<cycle>.wait-for-cycle](#), [<step>.wait-for-step](#)

datagram_link

Implemented By

[datagram_link_endpoint](#)

Description

```
SIM_INTERFACE(datagram_link) {
    /* Transmit a message to the object. */
    void (*receive)(conf_object_t *NOTNULL obj, bytes_t msg);
};

#define DATAGRAM_LINK_INTERFACE "datagram_link"
```

This interface is implemented by objects that receive messages from a datagram-link, and by the datagram-link connection endpoints themselves.

There is a single function *receive()*, which is used to pass the message *msg* to the object *obj*.

The message *msg* is treated as a series of bytes with no special format or meaning expected. If *obj* is a datagram-link endpoint, it will forward the message to all other endpoints registered on the link except the sender, effectively broadcasting the message on the link. If *obj* is a device, it will simply receive the message as sent by the original sender.

Note that the symmetry of the interface allows two devices to be connected directly to each other and talk as if connected via a datagram-link. This is however not supported by the default datagram-link component, so a special connector must be created for this purpose. Additionally, the standard link features, such as multicell configurations and latency, will not be available in that setup.

Execution Context

receive Execution context

ethernet_cable

Implemented By

[etg](#), [eth-cable-link-endpoint](#)

Description

```
SIM_INTERFACE(ethernet_cable) {
    void (*link_status)(conf_object_t *NOTNULL ep, bool link_up);
}
#define ETHERNET_CABLE_INTERFACE "ethernet_cable"
```

This interface is implemented by ethernet devices and link endpoints that are interested in the link status of the peer.

The *link_status* function is used to notify the peer that the link status at the local end is up or down. The **ethernet-cable-link** class propagates this information to the device connected at the other end.

Execution Context

link_status Execution context

ethernet_common

Implemented By

[BCM5703C](#), [BCM5704C](#), [DEC21041](#), [DEC21140A](#), [DEC21143](#), [dm9161](#), [etg](#), [eth-cable-link-endpoint](#), [eth-hub-link-endpoint](#), [eth-probe-port](#), [eth-switch-link-endpoint](#), [eth-injector](#), [generic_eth_phy](#), [i82543](#), [i82546](#), [mii-transceiver](#), [rapidio_link_endpoint](#), [rn-eth-bridge-raw](#), [rn-eth-bridge-tap](#), [rn-eth-proxy-raw](#), [rn-ip-router-raw](#), [service-node-device](#)

Description

```
SIM_INTERFACE(ethernet_common) {
    void (*frame)(conf_object_t *NOTNULL obj, const frags_t *frame,
                  eth_frame_crc_status_t crc_status);
};

#define ETHERNET_COMMON_INTERFACE "ethernet_common"
```

This interface is implemented by objects that receive Ethernet frames, both Ethernet devices and ethernet link endpoints.

There is a single function *frame* which sends an Ethernet frame, without preamble nor SFD (Start Frame Delimiter), but with a CRC field.

The *crc_status* parameter provides out-of-band information on the contents of the frame with regards to the CRC field using one of the values in the *eth_frame_crc_status_t* enum:

- *Eth_Frame_CRC_Match* means that the frame contents are correct. The CRC field in the frame should not be relied upon as its computation may have been skipped for optimization, and it may contain any value, including zero, a random value or a correctly computed CRC.
- *Eth_Frame_CRC_Mismatch* means that the frame contents are incorrect. The CRC field in the frame must contain a CRC that does not match the frame contents, i.e., to send an incorrect frame on the link, you must make sure that the CRC field will not match when computed.
- *Eth_Frame_CRC_Unknown* means that the relation between the frame contents and the CRC field is unknown. The relation can be established by computing the frame's CRC and comparing it to the frame's CRC field.

```
typedef enum {
    Eth_Frame_CRC_Match,
    Eth_Frame_CRC_Mismatch,
    Eth_Frame_CRC_Unknown
} eth_frame_crc_status_t;
```

When a device calls a link's *frame* function, it can set *crc_status* to any of the three values. If the link receives a *Eth_Frame_CRC_Unknown*, it will compute the CRC itself to set the status to *Eth_Frame_CRC_Match* or *Eth_Frame_CRC_Mismatch*.

When a link calls a device's `frame` function, `crc_status` will be set to either `Eth_Frame_CRC_Match` or `Eth_Frame_CRC_Mismatch`, and never `Eth_Frame_CRC_Unknown`.

When two devices are directly connected to each others without using a link, the interpretation of `Eth_Frame_CRC_Unknown` is up to the devices' implementation.

Execution Context

frame Execution context

ethernet_device

Implemented By

[BCM5703C](#), [BCM5704C](#), [DEC21041](#), [DEC21140A](#), [DEC21143](#), [etg](#), [i82543](#), [i82546](#), [mii-transceiver](#), [rn-eth-bridge-raw](#), [rn-eth-bridge-tap](#), [rn-eth-proxy-raw](#), [rn-ip-router-raw](#), [service-node-device](#)

Description

```
SIM_INTERFACE(ethernet_device) {
    void (*receive_frame)(conf_object_t *dev,
                          conf_object_t *link,
                          dbuffer_t *frame,
                          int crc_calculated);
    phy_speed_t (*auto_neg_request)(conf_object_t *dev,
                                    phy_speed_t speed);
    void (*auto_neg_reply)(conf_object_t *dev,
                           phy_speed_t speed);
};

#define ETHERNET_DEVICE_INTERFACE "ethernet_device"
```

Note: This interface is used only for old-style ethernet-links, which are obsolete. New devices should use the new-style Ethernet links such as **ethernet-switch**, **ethernet-cable**, or **ethernet-hub**, and the **ethernet_common** interface.

This interface is implemented by Ethernet device objects that connect to **ethernet-link** objects. It is used by the link object to send messages to the device object.

The *receive_frame* function is called when a frame has been sent by another device on the link. The frame is passed as a *dbuf* pointer that may not be modified without cloning it first. The *crc_calculated* parameter indicates whether the last four bytes of the frame contains a CRC. If it is false, the last four bytes should be assumed to contain a correct CRC, and if the actual CRC is needed, the receiving device must calculate it. If it is true, it is up to the receiving device to check if the CRC is correct or not.

The frame may be addressed to another device, which means that the device must be prepared to drop frames addressed to other devices, even if it has registered a MAC address. Also, the frame must not assume that it will receive all frames on the link if it has registered a MAC address, unless it tells the link it is running in promiscuous mode.

The *auto_neg_request* might be called when another device has requested auto-negotiation. The *speed* parameter contains the connection speeds that the other device supports. The return value should result of clearing the bits in *speed* that the device doesn't support.

The *auto_neg_reply* is called to return the result of a previous call to *auto_negoiate* in the **ethernet_link** interface.

Execution Context

<i>receive_frame</i>	Execution context
<i>auto_neg_request</i>	Execution context
<i>auto_neg_reply</i>	Execution context

ethernet_link

Implemented By
[ethernet-link](#)

Description

```
SIM_INTERFACE(ethernet_link) {
#ifndef GULP
    int (*connect_device)(conf_object_t *_obj, conf_object_t *dev,
                          int *new_connection);
    void (*disconnect_device)(conf_object_t *_obj,
                              conf_object_t *dev);
#endif
    void (*send_frame)(conf_object_t *_obj, int id,
                       dbuffer_t *frame,
                       int crc_calculated, nano_secs_t delay);
    void (*auto_negotiate)(conf_object_t *_obj, int id,
                           phy_speed_t speed);
    void (*add_mac)(conf_object_t *_obj, int id,
                    byte_string_t addr);
    void (*add_mac_mask)(conf_object_t *_obj, int id,
                         byte_string_t addr, byte_string_t mask);
    void (*delete_mac)(conf_object_t *_obj, int id,
                       byte_string_t addr);
    void (*delete_mac_mask)(conf_object_t *_obj, int id,
                           byte_string_t addr, byte_string_t mask);
    void (*clear_macs)(conf_object_t *_obj, int id);
    void (*promiscuous_mode)(conf_object_t *_obj, int id, int enable);
};

#define ETHERNET_LINK_INTERFACE "ethernet_link"
```

Note: This interface is used only for old-style ethernet-links, which are obsolete. New devices should use the new-style Ethernet links such as **ethernet-switch**, **ethernet-cable**, or **ethernet-hub**, and the **ethernet_common** interface.

This interface is implemented by Ethernet link objects that provide a data link layer interface for Ethernet frame delivery.

An Ethernet device calls the **connect_device** device function to attach itself to the link, and **disconnect_device** to detach itself. The device must implement the **ethernet_device** interface.

The **connect_device** function attaches an Ethernet device to the link. The device must implement the **ethernet_device** interface. The return value is an identification number that should be used in subsequent calls to the link to identify the device.

In addition, the *new_connection* parameter is used to return an extra value: if this value is true, this is a new connection, and the device should proceed to register its MAC address(es) with *add_mac* and *add_mac_mask*; but if false, this is an existing connection (restored from a checkpoint, for example), and the link already knows about the device's MACs.

The *disconnect_device* function detaches an Ethernet device from the link. It will not receive any more frames from the link and may not call any functions in the interface, except *connect_device*.

The *send_frame* function is used by a device to send an Ethernet frame onto the link to be delivered to the other devices connected to the same link. The frame should be a *dbuffer_t* containing a complete Ethernet frame, excluding the preamble and SFD, but including the CRC. The *crc_calculated* flag indicates whether the CRC is actually calculated. In many cases, the CRC will not be checked by the receiver, which makes it more efficient to skip the CRC calculation and set this flag to 0. If needed, Simics will calculate a valid CRC for the frame. The *delay* makes it possible to add a small delay to the frame. This can be used when a device wants to send multiple frames at once, but want them to be delivered in a specific sequence. Instead of using an event handler to send each frame, they can be sent at once, with an increasing delay for each frame. The delay is given in nanoseconds.

The *auto_negotiate* is used to do auto-negotiation of speed. An Ethernet device should call this function with the *speed* argument set to a value where all the bits that corresponds to speeds that the device can handle set. The device should call this function after it connects to the link the very first time (i.e. not after restarting from a checkpoint), or whenever the *auto_neg_request* function in its *ethernet_device* is called.

The *add_mac* function registers a MAC address that the device will accept frames for. By default, the device will receive no frames at all, but by calling this function, the device can inform the link that it will receive frames that match any of the MAC addresses it has registered. The device is not guaranteed to receive any frames not addressed to one of the registered addresses. The address is given as a string of six bytes in a *byte_string_t* structure.

The *add_mac_mask* is similar to *add_mac*, but in addition to the MAC address, a six byte bit mask is also be provided. This mask will be used to mask addresses when checking if they should be delivered to the device. Only bits in the address that has the corresponding bit in the mask set to 1 will be considered a match. If the bit mask is zero bytes, it will be handled as if it had been all ones, only an exact match will be accepted.

Most network cards listen to the broadcast address (ff:ff:ff:ff:ff:ff) or multicast addresses (01:00:00:00:00:01/01:00:00:00:00:00) in addition to the configured MAC address. Such models need to call *add_mac_mask* with appropriate arguments after being connected to a link.

The *delete_mac* and *delete_mac_mask* functions unregisters a MAC address previously registered with *add_mac* or *add_mac_mask*.

The *clear_macs* unregisters all previously registered MAC addresses.

The *promiscuous_mode* function sets the promiscuous flag that indicates that the device is listening to all MAC addresses. Setting this to true (non-zero) has a similar effect as *clear_macs* in that the link will start sending all frames to the device, but when the flag is set to false (zero) the MAC addresses filtering will resume with the previously registered MAC addresses.

The functions *auto_negotiate*, *add_mac*, *add_mac_mask*, *delete_mac*, *clear_macs*, and *promiscuous* behave slightly differently when called during initialization and during simulation. When loading a configuration, the devices may call these functions to set the initial configuration, but after the configuration has been set, the effect of these calls will be delayed by at least the latency of the link, since it is a change in the simulated state that needs to be propagated in a deterministic way.

Execution Context

<i>connect_device</i>	Outside execution context
<i>disconnect_device</i>	Outside execution context
<i>send_frame</i>	Instruction context
<i>auto_negotiate</i>	Instruction context
<i>add_mac</i>	Instruction context
<i>add_mac_mask</i>	Instruction context
<i>delete_mac</i>	Instruction context
<i>delete_mac_mask</i>	Instruction context
<i>clear_macs</i>	Instruction context
<i>promiscuous_mode</i>	Instruction context

ethernet_probe

Implemented By
[eth-probe](#)

Description

```
typedef enum {
    Eth_Probe_Port_A = 0,
    Eth_Probe_Port_B = 1
} eth_probe_side_t;

typedef void (*ethernet_probe_snoop_t)(lang_void *user_data,
                                         conf_object_t *probe,
                                         eth_probe_side_t to_side,
                                         const frags_t *frame,
                                         eth_frame_crc_status_t crc_status);

SIM_INTERFACE(ethernet_probe) {
    void (*attach_snooper)(conf_object_t *NOTNULL probe,
                           ethernet_probe_snoop_t snoop_fun,
                           lang_void *user_data);
    void (*attach_probe)(conf_object_t *NOTNULL probe,
                         ethernet_probe_snoop_t snoop_fun,
                         lang_void *user_data);
    void (*detach)(conf_object_t *NOTNULL probe);
    void (*send_frame)(conf_object_t *NOTNULL probe,
                       eth_probe_side_t to_side,
                       const frags_t *frame,
                       eth_frame_crc_status_t crc_status);
};

#define ETHERNET_PROBE_INTERFACE "ethernet_probe"
```

This interface is implemented by **eth-probe** objects. Once a probe has been inserted between a device and an Ethernet link, the functions of this interface can be used to setup callbacks:

attach_snooper()

Attach a snooper function: the probe will pass each frame to the snooper function, then forward it unchanged where it should be going

attach_probe()

Attach a probe function: the probe will pass each frame to the probe function, and give it the responsibility of forwarding the frame or any number of modified or additional frames using the *send_frame()* function.

detach

Detach the currently registered callback from the probe.

send_frame

Send a frame via the probe, either to the side A or B of the probe. Which side is which can be obtained with the *<eth-probe>.info* function.

This interface should only be used for inspection, and never as part of the actual simulation. The snoop functions must not affect the simulation in any way.

The *clock* parameter tells the link on which clock to post the events that call the snoop function. The snoop function will be called at the delivery time of the network packet, which means that it will be called at the same time as any ethernet devices attached to the same clock that receives packets from the same link.

Snooped frames with a matching CRC will contain the correct frame check sequence.

The *user_data* parameter is passed to the snoop function every time it is called.

Execution Context

<i>attach_snooper</i>	Instruction context
<i>attach_probe</i>	Instruction context
<i>detach</i>	Instruction context
<i>send_frame</i>	Instruction context

ethernet_snoop

Implemented By
[eth-cable-link](#), [eth-hub-link](#), [eth-switch-link](#)

Description

```
typedef void (*ethernet_link_snoop_t)(lang_void *user_data,
                                      conf_object_t *clock,
                                      const frags_t *packet,
                                      eth_frame_crc_status_t crc_status);

SIM_INTERFACE(ethernet_snoop) {
    conf_object_t *(*attach)(conf_object_t *NOTNULL link,
                           conf_object_t *clock,
                           ethernet_link_snoop_t snoop_fun,
                           lang_void *user_data);
};

#define ETHERNET_SNOOP_INTERFACE "ethernet_snoop"
```

This interface is implemented by ethernet link objects. It is used to attach snoop functions to the link. The snoop function will receive all traffic going over the link.

This interface should only be used for inspection, and never as part of the actual simulation. The snoop functions must not affect the simulation in any way.

The *clock* parameter tells the link on which clock to post the events that call the snoop function. The snoop function will be called at the delivery time of the network packet, which means that it will be called at the same time as any ethernet devices attached to the same clock that receives packets from the same link.

Snooped frames with a matching CRC will contain the correct frame check sequence. The *user_data* parameter is passed to the snoop function every time it is called.

Execution Context

attach Outside execution context

ethernet_vlan_snoop

Implemented By
[eth-switch-link](#)

Description

```
typedef void (*ethernet_link_snoop_t)(lang_void *user_data,
                                      conf_object_t *clock,
                                      const frags_t *packet,
                                      eth_frame_crc_status_t crc_status);

SIM_INTERFACE(ethernet_vlan_snoop) {
    conf_object_t *(*attach) (
        conf_object_t *NOTNULL link, conf_object_t *clock,
        ethernet_link_snoop_t snoop_fun, lang_void *user_data,
        uint16 vlan_id, bool is_vlan_trunk);
};

#define ETHERNET_VLAN_SNOOP_INTERFACE "ethernet_vlan_snoop"
```

This interface is implemented by ethernet VLAN switch link objects. It is used to attach snoop functions to the link. The snoop function will receive all traffic going over the link, either on a single VLAN, or on all of them.

This interface should only be used for inspection, and never as part of the actual simulation. The snoop functions must not affect the simulation in any way.

The *clock* parameter tells the link on which clock to post the events that call the snoop function. The snoop function will be called at the delivery time of the network packet, which means that it will be called at the same time as any ethernet devices attached to the same clock that receives packets from the same link.

Snooped frames with a matching CRC will contain the correct frame check sequence.

The *user_data* parameter is passed to the snoop function every time it is called.

The *vlan_id* indicates on which VLAN the snoop function should be attached (as its native VLAN).

The *is_vlan_trunk* flag indicates whether the snoop function should also receive the traffic on all other VLANs, tagged with an 802.1Q tag.

Execution Context

attach Outside execution context

extended_serial

Implemented By
[text-console](#)

Description

This interface extends the `serial_device` with a `write_at()` function. It is similar to the `write()` function of the mentioned interface, but accepts an on-screen character position. This interface is implemented by text consoles allowing them to be connected to text oriented frame buffers, such as VGA in text mode.

```
SIM_INTERFACE(extended_serial) {
    void (*write_at)(conf_object_t *obj,
                     int value, int x, int y, int fg, int bg);
    void (*graphics_mode)(conf_object_t *obj, int in_graphics_mode);
};

#define EXTENDED_SERIAL_INTERFACE "extended_serial"
```

Execution Context

Instruction Context for all methods.

firewire_bus

Implemented By
[firewire_bus](#)

Description

```
SIM_INTERFACE(firewire_bus) {
    int (*connect_device)(conf_object_t * NOTNULL bus,
                          conf_object_t * NOTNULL dev);
    int (*disconnect_device)(conf_object_t * NOTNULL bus,
                            conf_object_t * NOTNULL dev);
    void (*set_device_bus_id)(conf_object_t * NOTNULL bus,
                            conf_object_t * NOTNULL dev,
                            uint16 bus_id);

    void (*set_id_mask)(conf_object_t * NOTNULL bus,
                        conf_object_t * NOTNULL dev,
                        uint16 id_mask);

    firewire_ack_code_t (*transfer)(conf_object_t * NOTNULL bus,
                                    conf_object_t * NOTNULL source,
                                    dbuffer_t *packet, int crc_calculated);

    int (*register_channel)(conf_object_t * NOTNULL bus,
                           conf_object_t * NOTNULL dev,
                           uint32 channel);
    int (*unregister_channel)(conf_object_t * NOTNULL bus,
                            conf_object_t * NOTNULL dev,
                            uint32 channel);

    void (*reset)(conf_object_t * NOTNULL bus);
};

#define FIREWIRE_BUS_INTERFACE "firewire_bus"
```

This interface must be implemented by all firewire buses.

connect_device is called when a device wants to connect to the bus. The bus should return 0 upon success.

disconnect_device is called when a device wants to disconnect from the bus. The bus should return 0 upon success.

set_device_bus_id sets the bus id for the device. This needs to be called by a device when its Node_IDS[bus_id] field is updated. The new bus id should be placed in bits 15-6 in the bus_id argument.

set_id_mask can be called by a device to specify a mask that should be applied to the device ID when routing transfers. This can be used to e.g. accept transfers to multiple bus-numbers.

transfer sends a packet. The bus will route the transfer to the correct device(s). The *source* parameter should be set to the device which sent the transfer. The bus uses this parameter to avoid sending packets to their sender. If the *crc_calculated* argument is non-zero the packet's crc fields are filled in. The return code is one of:

```
typedef enum {
    /* Values defined in the FireWire specification */
    Firewire_Ack_Complete = 1,
    Firewire_Ack_Pending = 2,
    Firewire_Ack_Busy_X = 4,
    Firewire_Ack_Busy_A = 5,
    Firewire_Ack_Busy_B = 6,
    Firewire_Ack_Tardy = 0xb,
    Firewire_Ack_Conflict_Error = 0xc,
    Firewire_Ack_Data_Error = 0xd,
    Firewire_Ack_Type_Error = 0xe,
    Firewire_Ack_Address_Error = 0xf,

    /* Values not defined in FireWire but used in Simics */
    Firewire_Ack_No_Destination = 0x10, /* no destination found */
    Firewire_Ack_No_Ack = 0x11 /* no ack signal sent for packet */
} firewire_ack_code_t;
```

register_channel can be called to tell the bus that the device want to receive isochronous transfers to channel *channel*. *channel* must be in the range [0, 63). A device can register several channels. It is an error to subscribe a device to a channel it is already subscribed to. Returns 0 on success.

unregister_channel tells the bus the device is no longer interested in transfers to channel *channel*. *channel* must be in the range [0, 63). It is an error to unsubscribe a device from a channel if it isn't subscribing to it. Returns 0 on success.

reset can be called to cause a bus reset. After the bus reset, all the devices may have received new IDs.

Execution Context

Instruction Context for all methods.

firewire_device

Implemented By

[TSB12LV26](#), [firewire_device_test_wrapper](#), [firewire_sample_device](#)

Description

```
SIM_INTERFACE(firewire_device) {
    firewire_ack_code_t (*transfer)(conf_object_t * NOTNULL dev,
                                    dbuffer_t *packet, int crc_calculated);

    void (*reset)(conf_object_t * NOTNULL dev, uint16 id,
                  uint8 root_id,
                  uint32_array_t self_ids);
    uint32 (*get_self_id_template)();
    conf_object_t * NOTNULL dev);

    int (*get_rhb)(conf_object_t * NOTNULL dev);
    uint8 (*get_port_count)(conf_object_t * NOTNULL dev);
    uint16 (*get_port_mask)(conf_object_t * NOTNULL dev);
};

#define FIREWIRE_DEVICE_INTERFACE "firewire_device"
```

This interface must be implemented by all firewire devices.

transfer is called when the device receives a packet. If the *crc_calcualted* argument is non-zero the packet's *crc* fields are filled in. The return code is one of:

```
typedef enum {
    /* Values defined in the FireWire specification */
    Firewire_Ack_Complete = 1,
    Firewire_Ack_Pending = 2,
    Firewire_Ack_Busy_X = 4,
    Firewire_Ack_Busy_A = 5,
    Firewire_Ack_Busy_B = 6,
    Firewire_Ack_Tardy = 0xb,
    Firewire_Ack_Conflict_Error = 0xc,
    Firewire_Ack_Data_Error = 0xd,
    Firewire_Ack_Type_Error = 0xe,
    Firewire_Ack_Address_Error = 0xf,

    /* Values not defined in FireWire but used in Simics */
    Firewire_Ack_No_Destination = 0x10, /* no destination found */
    Firewire_Ack_No_Ack = 0x11 /* no ack signal sent for packet */
} firewire_ack_code_t;
```

reset is called when the bus is reset. *id* is the new ID of the device (top 10 bits represent the bus number, and the low 6 bits the node ID). *root_id* is the node ID for the root node. *self_ids* is the list of Self-ID packets during the Self-ID phase of the reset.

An array of unsigned 32-bit integers.

```
typedef struct {
    size_t len;
    uint32 *data;
} uint32_array_t;
```

get_self_id_template is an accessor function called by the bus or other self-ID provider to provide the static parts of the Self-ID for this device. The ID and port parts will be overwritten by the Self-ID provider.

get_rhb gets the current status of the root hold off bit in the device.

get_port_count returns the number of ports the device has. The largest allowed value is 16 and the lowest 1. This should remain constant during the lifetime of the device and always return a valid value.

get_port_mask returns a bitmask. Each bit is one if that port is enabled and zero if it is disabled. The least significant bit describes port 0 and the most significant bit port 15. The bits for ports which do not exist on the device should be set to zero.

Execution Context

Instruction Context for all methods.

frequency_listener

Implemented By
[clock](#), [frequency_bus](#)

Description

The `frequency_listener` interface is used for modeling frequency changes.

The frequency change initiator should call `set()` to set the new frequency. The `set()` function may be called multiple times with the same frequency value. The frequency is specified by the `numerator` and `denominator` arguments in units of Hz.

The `set()` function should also be used to set the initial value for a target.

An object that implements the `frequency_listener` interface typically has an attribute that refers to an object and port that implements the `simple_dispatcher` interface, from where the frequency originates.

The initiator of a frequency change should implement the `simple_dispatcher` interface, and it should only call the `set()` function on objects that have registered with this interface.

```
SIM_INTERFACE(frequency_listener) {  
    void (*set)(conf_object_t *NOTNULL obj, uint64 numerator,  
               uint64 denominator);  
};  
  
#define FREQUENCY_LISTENER_INTERFACE "frequency_listener"
```

Execution Context

Instruction Context for all methods.

generic_message_device

Implemented By

[generic-message-sample-device](#)

Description

```
SIM_INTERFACE(generic_message_device) {
    void (*receive_frame)(conf_object_t *dev, conf_object_t *link,
                          dbuffer_t *frame);
}
#define GENERIC_MESSAGE_DEVICE_INTERFACE "generic_message_device"
```

Note: This interface is used only for generic-links, which are obsolete. New link implementations should be based on the link library or the simple datagram-link example. Refer to the *Link Library Programming Guide* for more information.

This interface is implemented by generic message device objects that connect to **generic-message-link** objects. It is used by the link object to send messages to the device object.

The *receive_frame* function is called when a frame has been sent by another device on the link. The frame is passed as a `dbuffer_t` pointer that may not be modified without cloning it first.

Execution Context

receive_frame Execution context

generic_message_link

Implemented By
[generic-message-link](#)

Description

```
SIM_INTERFACE(generic_message_link) {  
    #ifndef GULP  
        int (*connect_device)(conf_object_t *_obj, conf_object_t *dev,  
                             int *new_connection, uint32 address);  
        void (*disconnect_device)(conf_object_t *_obj, conf_object_t *dev);  
    #endif  
        void (*send_frame)(conf_object_t *_obj, int id, uint32 address,  
                           dbuffer_t *frame, nano_secs_t delay);  
    };  
#define GENERIC_MESSAGE_LINK_INTERFACE "generic_message_link"
```

Note: This interface is used only for generic-links, which are obsolete. New link implementations should be based on the link library or the simple datagram-link example. Refer to the *Link Library Programming Guide* for more information.

This interface is implemented by generic message link objects that provide a data link layer interface for frame delivery.

An generic message device calls the *connect_device* device function to attach itself to the link, and *disconnect_device* to detach itself. The device must implement the generic_message_device interface.

The *connect_device* function attaches a generic link device to the link. The device must implement the generic_message_device interface. The return value is an identification number that should be used in subsequent calls to the link to identify the device. The *address* parameter sets the address of the device on the link.

The *disconnect_device* function detaches a generic link device from the link. It will not receive any more frames from the link and may not call any functions in the interface, except *connect_device*.

The *send_frame* function is used by a device to send a generic device frame onto the link to be delivered to another device connected to the same link. The frame should be a *dbufbuffer_t* containing a data frame. The *address* parameter is the address to sent the frame to. The *delay* makes it possible to add a small delay to the frame. This can be used when a device wants to send multiple frames at once, but want them to be delivered in a specific sequence. Instead of using an event handler to send each frame, they can be sent at once, with an increasing delay for each frame. The delay is given in nanoseconds.

Execution Context

<i>connect_device</i>	Outside execution context
<i>disconnect_device</i>	Outside execution context
<i>send_frame</i>	Instruction context

i2c_bridge

Implemented By

[i2c-bus](#)

Description

```
SIM_INTERFACE(i2c_bridge) {
    void (*address_added)(conf_object_t *device,
                          uint32 addr, uint32 mask);
    void (*address_removed)(conf_object_t *device,
                           uint32 addr, uint32 mask);
};

#define I2C_BRIDGE_INTERFACE "i2c_bridge"
```

This interface is implemented by any device that is registered in an I2C link with the *register_bridge* function. *register_bridge* will call *address_added*, possibly multiple times, to tell the bridge which slave addresses that are currently registered in the link. Until the bridge device disconnects from the link (using *disconnect_device*), the link will continuously call the functions *address_added* and *address_removed* whenever a slave device registers or unregisters from the link. In addition, when an i2c bridge is disconnected from a link, the *address_removed* function is called for all currently registered slave addresses.

The *addr* and *mask* parameters are interpreted in the same way as in the function *register_slave_address* in the *i2c_link* interface.

The *i2c_bridge* interface is mainly useful when implementing bridges between I2C links: When a bridge forwards traffic from one link to another, it needs the information provided via this interface to correctly forward all relevant traffic between the links, while avoiding to register twice to the same address.

Execution Context

Instruction Context for all methods.

i2c_bus

Implemented By

[i2c-bus](#)

Description

```
SIM_INTERFACE(i2c_bus) {
    int (*start)(conf_object_t *i2c_bus, uint8 address);
    int (*stop)(conf_object_t *i2c_bus);
    uint8 (*read_data)(conf_object_t *i2c_bus);
    void (*write_data)(conf_object_t *i2c_bus, uint8 value);
    int (*register_device)(conf_object_t *i2c_bus,
                           conf_object_t *device,
                           uint8 address, uint8 mask,
                           i2c_device_flag_t flags);
    void (*unregister_device)(conf_object_t *i2c_bus,
                             conf_object_t *device,
                             uint8 address, uint8 mask);
};

#define I2C_BUS_INTERFACE "i2c_bus"
```

The `i2c_bus` interface is implemented by the I2C bus. The interface is used by I2C devices to communicate with the I2C bus.

To connect an I2C device to an I2C bus, first you call `register_device` with a 7-bit `address` and `mask`. The address is actually an address pattern. When there is traffic on the I2C bus (as initiated by a call to the bus interface `start` function), the target address is matched against each registered device by checking if `(target_address ^ device_address) & device_mask == 0`. The `flags` attribute sets the type of connection. The alternatives are `exclusive` or `shared`. An I2C device connected `exclusive` can not share a transfer with another I2C device. An I2C device connected `shared` supports that other I2C devices connected `shared` to the I2C bus handles the same transfer. The `register_device` function returns 0 on success.

Use `unregister_device` to disconnect an I2C device from the I2C bus. To completely remove a device use the same `address` and `mask` attributes as when the device was registered. An I2C device can also remove some part of the address match by unregister itself with a different mask.

An I2C transfer is initiated by a master I2C device. The I2C device responding to the transfer is called slave. The master starts a transfer by calling the `start` function with `address` as argument. The `address` is the 7-bit address plus a read/write bit (read/write = 0 → slave-receive, read/write = 1 → slave-transmit). The read/write bit is the least significant bit. This means that all odd values sent to `start` function initiates a transfer where the master is requesting data from the slave.

I2C devices implement the `i2c_device` interface. Both the `i2c_device` interface and the `i2c_bus` interface has identical `read_data` and `write_data` functions to transfer data over the bus. The `i2c_device` also has `set_state` function, which is used by the I2C bus to set the I2C device state. The states are `I2C_idle`, `I2C_slave_transmit`, and `I2C_slave_receive`. The default state is `I2C_idle`.

Here are the steps for a transfer:

- 1 . The I2C master device calls `start` in the I2C bus with 7-bit address and 1-bit read/write flag.
- 2 . The I2C bus calls the `set_state` in the I2C slave device with `I2C_slave_transmit` or `I2C_slave_receive` as argument depending on the 1-bit read/write flag. The I2C slave accepts the state change by returning 0.
- 3 . I2C bus returns 0 to the I2C master if the start command in 1 was successful. The I2C bus can report failure for several reasons, there are another ongoing transfer, can not find any device with the address provided, I2C slave did not except state change etc.
- 4 . The I2C master calls the I2C buses `read_data` or `write_data` depending if it wants to read from or write to the I2C slave. The I2C bus relays the call to the I2C slave `read_data` or `write_data` function. The I2C slave have no way to report errors, the I2C master expects the I2C slave to be able to handle all calls without any problem. The I2C bus can do step 4 several times before terminating the transfer.
- 5 . The I2C master calls the `stop` function when it wants to terminate the transfer. This causes the I2C bus to call the `set_state` function in the I2C slave with `I2C_idle` as argument. The transfer is now completed.

Execution Context

Instruction Context for all methods.

i2c_device

Implemented By

[AT24Cxx](#), [PCF8582C](#), [i2c_slave_v2_to_bus_adapter](#)

Description

```
SIM_INTERFACE(i2c_device) {
    int (*set_state)(conf_object_t *device, i2c_device_state_t state,
                     uint8 address);
    uint8 (*read_data)(conf_object_t *device);
    void (*write_data)(conf_object_t *device, uint8 value);

};

#define I2C_DEVICE_INTERFACE "i2c_device"
```

The `i2c_device` interface is implemented by I2C devices. The `set_state` function is used to change the state of the I2C device. The I2C device accepts the state change by returning 0. The I2C device must accept calls to `read_data` when its state is `I2C_slave_transmit` and calls to `write_data` when its state is `I2C_slave_receive`. See the description for the `i2c_bus` interface for more information how to use this interface.

Execution Context

Instruction Context for all methods.

i2c_link

Implemented By

[i2c_link_v1](#)

Description

```
SIM_INTERFACE(i2c_link) {
    void (*register_slave_address)(conf_object_t *i2c_link,
                                    conf_object_t *slave,
                                    uint32 address, uint32 mask);
    void (*unregister_slave_address)(conf_object_t *i2c_link,
                                    conf_object_t *slave,
                                    uint32 address, uint32 mask);

    void (*register_bridge)(conf_object_t *i2c_link,
                           conf_object_t *bridge);

    void (*disconnect_device)(conf_object_t *i2c_link,
                             conf_object_t *device);

    void (*start_request)(conf_object_t *i2c_link, conf_object_t *master,
                         uint32 address);
    void (*read_request)(conf_object_t *i2c_link, conf_object_t *master);
    void (*ack_read_request)(conf_object_t *i2c_link, conf_object_t *master,
                            i2c_status_t ack);
    void (*write_request)(conf_object_t *i2c_link, conf_object_t *master,
                         uint8 value);

    void (*read_response)(conf_object_t *i2c_link, conf_object_t *slave,
                         uint8 value);
    void (*ack_read_response)(conf_object_t *i2c_link,
                             conf_object_t *slave);
    void (*write_response)(conf_object_t *i2c_link, conf_object_t *slave,
                          i2c_status_t status);
    void (*start_response)(conf_object_t *i2c_link, conf_object_t *slave,
                          i2c_status_t status);
    void (*stop)(conf_object_t *i2c_link, conf_object_t *master);
};

#define I2C_LINK_INTERFACE "i2c_link"
```

The `i2c_link` interface is implemented by the I2C link. The interface is used by I2C devices to communicate with the I2C link.

An I2C device implements either the `i2c_slave` or the `i2c_master` interface, or both. You don't need to do anything to connect an I2C master device to a link.

To connect an I2C slave device to an I2C link, call `register_slave_address` with an *address pattern* consisting of the `address` and `mask` parameters. When there is traffic on the I2C link (as initiated by a call to the link interface `start_request` function), the target address is matched against each registered device by checking if `(target_address ^ device_address) & device_mask == 0`; i.e., the bits that are set to 1 in the mask indicates which bits in the address that must match.

There are three different *addressing modes* defined by this protocol: *7-bit addressing*, *General call*, and *10-bit addressing*. 7-bit addressing is by far the most common addressing mode, and at the moment the only one supported by the official `i2c_link` device. When 7-bit addressing is used, the address fits into the lower 8 bits of the `address` parameter; in other addressing modes more than 8 bits can be used, but the addressing mode can always be deduced from the lower 8 address bits.

In 7-bit addressing mode, the address of an I2C device is encoded as an even 8-bit number, in the range `0x10 - 0xef`. In other words, address ranges `0x00 - 0x0f` and `0xf0 - 0xff` are reserved for other addressing modes, and the 7-bit address is defined by bits 7 to 1 (little-endian) of the 8-bit encoded address. Bit 0 is reserved as a read/write bit for the `start_request` function, and should be 0 in both `address` and 0 in `mask`, while bits 8 to 31 should be 0 in `address` and 1 in `mask`.

It is an error to register two I2C slave devices to the same 7-bit address.

Use `disconnect_device` to completely disconnect an I2C slave device from the I2C link. If you just intend to shut disable the I2C functionality of the device (without disconnecting the wire), use `unregister_slave_address`, and use the same `address` and `mask` attributes as when the device was registered. An I2C slave device can also remove some part of the address match by unregistering itself with a different mask.

An I2C transfer is initiated by a master I2C device. The I2C device responding to the transfer is called slave. The master starts a transfer by calling the `start_request` function with `address` as argument. In 7-bit addressing mode, the `address` is the 7-bit address plus a read-write bit (read-write = 0 → slave-receive, read-write = 1 → slave-transmit). The read-write bit is the least significant bit. This means that all odd values sent to `start` function initiates a transfer where the master is requesting data from the slave.

I2C slave devices implement the `i2c_slave` interface. The `i2c_slave` interface and the `i2c_link` interface have identical `start_request`, `read_request`, `ack_read_request` and `write_request` functions to transfer data over the link. See the `i2c_slave` interface for the definitions of these functions.

I2C master devices implement the `i2c_master` interface, which has the functions `start_response`, `read_response`, `ack_read_response` and `write_response` in common with the `i2c_link` interface. The only difference in definition is that the `I2C_STATUS_BUS_BUSY` value is not allowed for the `status` parameter of `start_response` in the `i2c_link` interface, while it is allowed in the `i2c_master` interface. The definitions of these functions can be found in the documentation of the `i2c_master` interface.

Here are the steps for a transfer:

1. The I2C master device calls *start_request* in the I2C link with 7-bit address and 1-bit read/write flag.
2. The I2C link calls *start_request* in the I2C slave device, forwarding the address given by the master. If there is no slave device on the given address, or if the link is busy with another connection, the I2C link instead terminates the connection by calling *start_response* in the master device with *I2C_status_success* or *I2C_status_bus_busy* as status code.
3. The I2C slave device calls *start_response* in the I2C link, with either *I2C_status_success* or *I2C_status_noack* as status code.
4. The I2C link calls *start_response* in the I2C master device. This terminates the transfer if the status wasn't *I2C_status_success*.
5. The I2C master calls the *read_request* or *write_request* function in the I2C link, depending on the read/write bit given in the *start_request* message. The I2C link relays the call to the I2C slave *read_request* or *write_request* function. The slave responds with a call to *read_response* or *write_response*, which the link relays to the I2C master device. If it's a read transfer, the master calls the *ack_read_request* function in the I2C link, which is relayed to the slave. The slave responds with a call to the *ack_read_response* in the I2C link, which is relayed to the master. This step can be repeated any number of times.
6. The I2C master ends the transfer either by calling *start_request* again (possibly with a different slave address), or by calling *stop* in the I2C link. The link will respond by calling *stop* in the slave device (if it wasn't a repeated start with the same device as slave), and if it was a start command, relay it to the slave device as in step 2 above. Before a master device is allowed to call *start_request* or *stop*, it must wait for any pending *read_response* or *write_response* call from a slave device. In a read transfer, the last call to the function *ack_read_request* should pass *I2C_status_noack* in the *ack* parameter.

In total, an I2C link can be in seven different states:

1. Idle, waiting for *start_request* call from any master device
2. Waiting for a *start_response* call from the slave device (write mode)
3. Waiting for a *write_request* (or *stop* or repeated *start_request*) call from the current master device
4. Waiting for a *write_response* call from the slave device
5. Waiting for *start_response* call from the slave device (read mode)
6. Waiting for a *read_request* (or *stop* or repeated *start_request*) call from the current master device
7. Waiting for a *read_response* call from the slave device
8. Waiting for an *ack_read_request* call from the current master device
9. Waiting for an *ack_read_response* call from the slave device

During a transfer, the actual data is delivered by the *write_request* and *read_response* function calls. The data is always delivered one byte at a time.

There are two addressing modes apart from the 7-bit addressing mode described above:

- General call address: This is a kind of broadcast signal; an I2C master device writes data to the link, that any I2C device can listen to. A General call is issued by calling *request_start* with *address* = 0. To listen to General call, call *register_slave_address* with *addr* = 0 and *mask* = 0xffffffff. Multiple slave devices can register to the General call address; calls to *start_request* and *write_request* will be relayed to each registered slave device. The *start_response* and *write_response* functions in the calling master's *i2c_master* interface will be called when *all* slave devices have called the corresponding function in the *i2c_link* interface.
- 10-bit addresses: This is an extension to the protocol, where two bytes are sent as address, of which 10 bits are interpreted as address bits. The address sent to *register_slave_address* consists of the two bytes that are sent, with the first sent byte as the least significant byte. This means that the bits of the *address* parameter are defined as follows:
 - Bit 0 = 0
 - Bits 1-2 correspond to bits 8-9 of the 10-bit address.
 - Bits 3 = 0
 - Bits 4-7 = all 1
 - Bits 8-15 correspond to bits 0-7 of the 10-bit address.
 - Bits 16-31 = all 0

Similarly, the *mask* parameter should have bit 0 set to 0 and bits 3-7 and 16-31 set to 1.

A transmission from a master to a 10-bit addressed slave must start with a write transaction, which optionally is followed by a read transaction from the same slave. The write transaction is started by a *start_request* call, whose *address* parameter is a 16-bit number, composed in the same way as the *address* parameter of *register_slave_address* described above, with bit 0 set to 0. After writing the desired number of bytes, a repeated start command can be issued to start the read transaction to the same slave device. The repeated start command is issued by calling *start_request*, now with an 8-bit *address* parameter, which consists of the lower 8 bits from the *address* parameter of the previous *start_request* call, with bit 0 flipped to 1.

It is an error to register two I2C slave devices to the same 10-bit address.

Note that a single call to *register_slave_address* may only register addresses from *one* addressing mode; it is an error to supply an address pattern that matches addresses from different addressing modes.

During write operations, ACK bits are sent in the *status* parameter of the *write_response* functions. During read operations, the master device sends the ACK bit with the *ack_read_request* function after the *read_response* function has been called by the slave. The

slave should respond with a call to the *ack_read_response* function, which is relayed to the master. The value of the ACK bit controls which functions that are allowed as the next operation by the master device: If the ACK bit is 0 (i.e., `I2C_STATUS_SUCCESS`), the next operation must be a *read_request*, while `ACK = 1` (`I2C_STATUS_NOACK`) means that the next operation must be *stop* or a repeated *start_request*. It is an error to violate this rule: If the master device's software attempts to issue a STOP or START condition after a read with `ACK=0`, the master device should catch the error and log a `specViolation` message.

The *register_bridge* function is used to register a device that implements the `i2c_bridge` interface. The I2C link device will use the `i2c_bridge` interface to keep the I2C bridge device updated with which slave addresses on the link that have something registered on them; this is mainly useful when implementing a bridge between i2c links. For more information, see the documentation on the `i2c_bridge` interface.

The *disconnect_device* function completely disconnects a device (slave, master, bridge or any combination thereof) from the link. The link checks which i2c interfaces the device implements, and cleanly removes all its references to the device. This mainly has following effects:

- The link is prevented from issuing pending *response*-like calls to a disconnected master or bridge device
- If the device is currently involved in a data transfer, the transfer will be aborted as cleanly as possible.
- If the device was registered with the *register_bridge* function, its *address_removed* function is called for each currently registered i2c slave address.
- The disconnected device is erased from any internal caches; this prevents certain memory corruption errors which can occur if one device is destroyed and another one is created on exactly the same memory location.

Hence, *disconnect_device* an I2C device which has had any kind of interaction with an I2C link, must always use *disconnect_device* when disconnecting.

Execution Context

Instruction Context for all methods.

i2c_master

Implemented By

[i2c-bus](#)

Description

```
SIM_INTERFACE(i2c_master) {
    void (*bus_freed)(conf_object_t* device);
    void (*read_response)(conf_object_t *device, uint8 value);
    void (*ack_read_response)(conf_object_t *device);
    void (*write_response)(conf_object_t *device, i2c_status_t status);
    void (*start_response)(conf_object_t *device, i2c_status_t status);
};

#define I2C_MASTER_INTERFACE "i2c_master"
```

The `i2c_master` interface should be implemented by devices that intend to act as a master on an I2C link.

The functions `start_response`, `read_response`, `ack_read_response` and `write_response` are called as a response to the corresponding `request_*` calls to the I2C link.

The `status` parameter to the `start_response` function can have three values:

I2C_status_success

The transfer was successfully started

I2C_status_noack

No I2C slave device accepted the transfer, likely because there was no I2C slave device listening to the given address

I2C_status_bus_busy

An existing transfer blocks the I2C link. When the existing transfer is completed (i.e., on the next STOP condition), the link will call the `bus_freed` function in the master device, if it's implemented. This makes it possible to avoid periodical polling of a busy link.

The `status` parameter to the `write_response` function can have two values, `I2C_status_success` for success, or `I2C_status_noack` if the slave device did not ack the written byte.

Execution Context

Instruction Context for all methods.

i2c_master_v2

Implemented By

[i2c-link-endpoint](#), [i2c_link_v1](#), [i2c_slave_v2_to_bus_adapter](#), [i2c_wire](#)

Description

```
typedef enum {
    I2C_ack = 0,
    I2C_noack = 1
} i2c_ack_t;

SIM_INTERFACE(i2c_master_v2) {
    void (*acknowledge)(conf_object_t *device, i2c_ack_t ack);
    void (*read_response)(conf_object_t *device, uint8 value);
};

#define I2C_MASTER_V2_INTERFACE "i2c_master_v2"
```

The `i2c_master_v2` interface should be implemented by devices that intend to act as a master on an I2C link.

The function `acknowledge` is called in response to a `start` or `write` call in the `i2c_slave_v2` interface of the slave device. `read_response` is called as response to a `read` call. More details can be found in the documentation of the [i2c_slave_v2](#) interface.

Execution Context

Instruction Context for all methods.

i2c_slave

Implemented By

[i2c-bus](#)

Description

```
SIM_INTERFACE(i2c_slave) {
    void (*start_request)(conf_object_t *device, uint32 address);
    void (*read_request)(conf_object_t *device);
    void (*ack_read_request)(conf_object_t *device, i2c_status_t ack);
    void (*write_request)(conf_object_t *device, uint8 value);
    void (*stop)(conf_object_t *device, uint8 repeated_start);
};

#define I2C_SLAVE_INTERFACE "i2c_slave"
```

The `i2c_slave` interface is implemented by devices that may act as slaves on an I2C link. The `start_request` function is called by the I2C link to start a transaction. Bits 7 downto 1 of the `address` parameter are the address of the slave, and bit 0 is the write bit. The started transaction is a write transaction if the write bit is 0, and a read transaction otherwise. The I2C slave device accepts the transaction by calling the `start_response` function in the I2C link. The transaction consists of a number of calls to `write_request` (or `read_request`, depending on the write bit). The slave responds to each of these with a call to `write_response` (or `read_response`) in the I2C link. The `ack_read_request` function is called by the I2C link as response to the `read_response` function. The transfer can end in three ways:

1. If a `stop` condition occurs (i.e., a master calls `stop`), `stop(0)` is called in the slave device.
2. If a `repeated start` condition occurs, with the same device as slave, then the start message is relayed to the slave device as usual, and the first transmission is ended by this new call to the slave's `start_request` function.
3. If a `repeated start` condition occurs, with a different device as slave, then the first transmission is ended by a `stop(1)` call in the first slave device.

Note that a call to `stop` doesn't necessarily represent an I2C stop condition: If the `repeated_start` parameter is 1, the call actually represents an I2C start condition, whose effect is similar to that of a stop condition.

Note that a call to `ack_read_request` always is followed by a call to `read_request` if `ack` is `I2C_status_success`, or by a call to either `stop` or `start_request` if `ack` is `I2C_status_noack`. Moreover, the interface calls to an i2c slave device always follow the regular expression $((R(ra)*rA) | (Ww*)) +P)*$, where `R` and `W` are calls to `start_request` with odd and even addresses, respectively; `r` is `read_request`; `w` is `write_request`;

`a` and `A` are `ack_read_request` calls with `ack` set to `I2C_status_success` and `I2C_status_noack`, respectively; and `P` is a call to `stop`.

See the description for the `i2c_link` interface for more information how to call the `*_response` functions.

Execution Context

Instruction Context for all methods.

i2c_slave_v2

Implemented By

[AT24Cxx](#), [PCF8582C](#), [i2c-link-endpoint](#), [i2c_link_v1](#), [i2c_wire](#)

Description

```
SIM_INTERFACE(i2c_slave_v2) {
    void (*start)(conf_object_t *device, uint8 address);
    void (*read)(conf_object_t *device);
    void (*write)(conf_object_t *device, uint8 value);
    void (*stop)(conf_object_t *device);
    attr_value_t (*addresses)(conf_object_t *device);
};

#define I2C_SLAVE_V2_INTERFACE "i2c_slave_v2"
```

The `i2c_slave_v2` interface is used together with the [i2c_master_v2](#) interface to model communication over an I2C bus. Most calls to a method in the `i2c_slave_v2` interface expect a response call to a method in the [i2c_master_v2](#) interface. It is up to each device to find the caller's interface; usually it is configured with an attribute in both the master and slave device.

The `i2c_slave_v2` and `i2c_master_v2` interfaces replace the obsolete interfaces `i2c_master`, `i2c_slave`, `i2c_link`, and `i2c_bridge`.

An I2C bus consists of a number of *master* devices and a number of *slave* devices. Each slave device listens to one or more 7-bit *addresses*. It is an error to connect two slave devices to the same bus if they listen to the same address. Communication over the bus is initiated by a master, which can communicate with one slave device at a time. Only one master on a bus can communicate at a time.

Two I2C devices can communicate directly if one implements the `i2c_master_v2` interface and the other one implements the `i2c_slave_v2` interface. An I2C bus with multiple masters or slaves is typically modeled using an `i2c-link-v2` object, where each device is connected to an endpoint object.

The *start* method starts a transfer. The *address* parameter is the address of the slave device, encoded as an 8-bit number. Using little-endian bit numbering, bit 0 is a read/write bit (0 → master writes, 1 → master reads) and bits 1 to 7 is the 7-bit address of the slave device. The *address* parameter should normally be in the (inclusive) range 0x10 – 0xef; the ranges 0x00 – 0x0f and 0xf0 – 0xff are reserved for special addressing modes.

The slave responds to *start* using the *acknowledge* method. The *ack* parameter may be `I2C_ack` or `I2C_noack` if the start is acked or noacked, respectively. In a multi-master configuration, the slave may instead respond by calling the *start* method in the `i2c_slave_v2` interface of the master device; this signals that the master lost the bus arbitration. This is discussed further below.

If a start was acked by a slave device, read/write bit in the *start* method was 1, the master proceeds with a sequence of *read* calls. Each call is followed by a *read_response* call from the slave. If the read/write bit in the *start* was 0, the master instead proceeds with a sequence of *write* calls, each one followed by an *acknowledge* call. The *ack* parameter of the *acknowledge* method may be either `I2C_ack` or `I2C_noack`, representing ack and noack, respectively.

After sending a number of reads or writes, the master may choose to either call the *stop* method to issue a *stop condition* which terminates the transfer, or to call the *start* method again to issue a *repeated start condition*. The repeated start condition works like a normal start condition.

When 0 is passed as the *address* parameter of the *start* method, a *General Call* transfer is initiated. A General Call transfer works like a normal write transfer, except that multiple slave devices may acknowledge the start and write requests. The master will conceive a request as acknowledged if it was acknowledged by at least one of the slaves.

The *addresses* method is used to retrieve the set of addresses a device currently listens to. The method is called by the `i2c-link-v2` to provide an overview of the bus topology, and to detect address collisions early. The method may not have any side-effects on the simulation. The return value is a list containing all the different values of the *address* parameter in a *start* call that the slave would respond to with `I2C_ack`. The list may also contain an element `Nil` to mark that the device might respond to other addresses as well. If a device is configured to listen to a specific address but is currently disabled, then that address may still be included in the list.

For example, if a slave device listens to the single 7-bit address 0x55, the return value of *addresses* would be `[0xaa, 0xab]`, using Python list syntax.

The specification of the return value may be extended in the future, to allow other values than `Nil` and 8-bit integers. Therefore, callers should not make any assumptions on how the elements are formatted in the return value.

For most I2C devices it is sufficient to implement only one of the `i2c_slave_v2` and `i2c_master_v2` interfaces. In some cases it may be useful for a device to implement both interfaces, either because the device can act both as a master and as a slave on the bus, or because it needs to use the *start* and *stop* methods to monitor start and stop conditions on the bus. I2C link endpoints also implement both interfaces.

If multiple master devices are connected to the same i2c link, then all master devices need to implement the `i2c_slave_v2` interface in addition to `i2c_master_v2`, in order to handle bus arbitration correctly:

- A master device should monitor calls to the *start* and *stop* functions: After a call to *start* in a master device, the master should not call *start* in the slave until after a subsequent call to *stop* in the master.
- When a master calls *start* in a link endpoint device, the endpoint may respond with a call to *start* instead of *acknowledge*; this means that there was bus arbitration, which the master lost to the given *start* request. Note that this case can

not happen on a latency-free connection, since all arbitration is handled by the master's monitoring of *start* and *stop* calls. Note also that user-written slave devices are required to respond with *acknowledge* to a *start* call; only the i2c link endpoint (and other i2c infrastructure objects from the Simics-Base package) are allowed to respond with a *start* call.

Note that there is no need for a master device to implement the `i2c_slave_v2` interface if the device only will be used in single-master configurations.

The *start*, *write* and *read* methods in the `i2c_slave_v2` interface are allowed to respond synchronously; i.e., the *acknowledge* or *read_response* method may be called before the corresponding method in the slave interface has returned. A master that needs to connect directly to a slave device needs to take this into consideration; however, when communicating via an `i2c-link-v2` object, the link guarantees that all responses are asynchronous.

Execution Context

addresses	Outside Execution Context
All other methods	Instruction Context

ieee_802_3_mac

Implemented By

[AM79C973](#), [DEC21140A-dml](#), [i82559](#)

Description

```
#define IEEE_802_3_MAC_INTERFACE "ieee_802_3_mac"
SIM_INTERFACE(ieee_802_3_mac) {
    int (*receive_frame)(conf_object_t *obj, int phy,
                         dbuffer_t *buf, int crc_ok);
    void (*tx_bandwidth_available)(conf_object_t *obj, int phy);
    void (*link_status_changed)(conf_object_t *obj, int phy,
                                ieee_802_3_link_status_t status);
};

typedef enum {
    IEEE_link_unconnected,
    IEEE_link_down,
    IEEE_link_up
} ieee_802_3_link_status_t;
```

Deprecated; use `ieee_802_3_mac_v3` instead.

Interface that should be implemented by 802.3 media access control layers.

The `receive_frame` function is called when a frame has been received by the phy. The frame is passed as a `dbufbuffer_t` that may not be modified without cloning it first. The return value have no meaning, callers should ignore it, and new implementations should return 0.

The `tx_bandwidth_available` is called by the PHY when a previous call to `send_frame` or `check_tx_bandwidth` in the `ieee_802_3_phy` interface have returned no bandwidth available.

`link_status_changed` is called when the phy detects a change of the link status.

The `phy` parameter is a number that identifies this particular PHY, in configurations with several PHYs connected to the same MAC.

Execution Context

Instruction Context for all methods.

ieee_802_3_mac_v3

Implemented By

[AM79C973](#), [DEC21140A-dml](#), [i82559](#), [mac_splitter](#)

Description

```
#define IEEE_802_3_MAC_V3_INTERFACE "ieee_802_3_mac_v3"
SIM_INTERFACE(ieee_802_3_mac_v3) {
    void (*receive_frame) (
        conf_object_t *obj, int phy, const frags_t *frame, int crc_ok);
    void (*tx_bandwidth_available)(conf_object_t *obj, int phy);
    void (*link_status_changed) (
        conf_object_t *obj, int phy, ieee_802_3_link_status_t status);
};

typedef enum {
    IEEE_link_unconnected,
    IEEE_link_down,
    IEEE_link_up
} ieee_802_3_link_status_t;
```

Interface that should be implemented by 802.3 media access control layers.

The *receive_frame* function is called when a frame has been received by the phy. The frame is passed as a *frags_t* that must not be modified.

The *crc_ok* flag is set to indicate that the frame is valid with regards to CRC calculations. If the *crc_ok* flag is not set, the frame is considered invalid. Note that in general, *crc_ok* does not indicate whether or not the CRC field in the frame can be relied upon, or whether its computation has been skipped to improve simulation performance. When using new-style links, *crc_ok* set to false indicates that the CRC field contains valid information that is not matching with the frame contents.

The *tx_bandwidth_available* is called by the PHY when a previous call to *send_frame* or *check_tx_bandwidth* in the *ieee_802_3_phy_v3* interface have returned no bandwidth available.

link_status_changed is called when the phy detects a change of the link status.

The *phy* parameter is a number that identifies this particular PHY, in configurations with several PHYs connected to the same MAC.

Execution Context

Instruction Context for all methods.

ieee_802_3_phy

Implemented By

[dm9161](#), [generic_eth_phy](#), [mii-transceiver](#)

Description

```
#define IEEE_802_3_PHY_INTERFACE "ieee_802_3_phy"
SIM_INTERFACE(ieee_802_3_phy) {

    int (*send_frame)(conf_object_t *obj, dbuffer_t *buf, int replace_crc);
    int (*check_tx_bandwidth)(conf_object_t *obj);

#ifndef !defined(GULP)
    void (*add_mac)(conf_object_t *obj, const uint8 *mac);
    void (*del_mac)(conf_object_t *obj, const uint8 *mac);
    void (*add_mac_mask)(conf_object_t *obj, const uint8 *mac,
                         const uint8 *mask);
    void (*del_mac_mask)(conf_object_t *obj, const uint8 *mac,
                         const uint8 *mask);
#endif
    void (*set_promiscous_mode)(conf_object_t *obj, int enable);

};
```

Interface that should be implemented by 802.3 physical layers. Deprecated; use [ieee_802_3_phy_v2](#) instead.

Execution Context

Undefined.

ieee_802_3_phy_v2

Implemented By

[dm9161](#), [generic_eth_phy](#), [mii-transceiver](#)

Description

```
#define IEEE_802_3_PHY_V2_INTERFACE "ieee_802_3_phy_v2"
SIM_INTERFACE(ieee_802_3_phy_v2) {
    int (*send_frame)(conf_object_t *obj, dbuffer_t *buf, int replace_crc);
    int (*check_tx_bandwidth)(conf_object_t *obj);
    void (*add_mac)(conf_object_t *obj, byte_string_t mac);
    void (*del_mac)(conf_object_t *obj, byte_string_t mac);
    void (*add_mac_mask)(conf_object_t *obj, byte_string_t mac,
                         byte_string_t mask);
    void (*del_mac_mask)(conf_object_t *obj, byte_string_t mac,
                         byte_string_t mask);
    void (*set_promiscous_mode)(conf_object_t *obj, int enable);
}
```

Deprecated; use `ieee_802_3_phy_v3` instead.

Interface that should be implemented by 802.3 physical layers.

The *send_frame* function is used by a device to send an Ethernet frame. The frame should be a `dbuf` containing a complete Ethernet frame, excluding the preamble and SFD, but including the CRC. The *replace_crc* flag indicates whether the CRC is not actually calculated yet. The passed *buf* should not be modified by the called function. If the function return 0, the frame was sent to the link; In case -1 is returned, there was not enough bandwidth available right now, and the frame could not be sent. The PHY should call the *tx_bandwidth_available* in the `ieee_802_3_mac` interface at the MAC, when the frame can be sent.

The *check_tx_bandwidth* can also be used to check that there is bandwidth available, without sending a frame. It returns 0 if there is no bandwidth available, and a positive value if the frame can be sent right away.

add_mac, *del_mac* and *set_promiscous_mode* have the same functionality as the equivalent functions in `etherlink` interface. They do nothing if the PHY is connected to the new-style links using the `ethercommon` interface.

The word “promiscuous” is misspelled in the interface.

Execution Context

Instruction Context for all methods.

ieee_802_3_phy_v3

Implemented By

[dm9161](#), [generic_eth_phy](#), [mii-transceiver](#)

Description

```
#define IEEE_802_3_PHY_V3_INTERFACE "ieee_802_3_phy_v3"
SIM_INTERFACE(ieee_802_3_phy_v3) {
    int (*send_frame) (
        conf_object_t *obj, const frags_t *frame, int replace_crc);
    int (*check_tx_bandwidth)(conf_object_t *obj);
};
```

Interface that should be implemented by 802.3 physical layers.

The *send_frame* function is used by a device to send an Ethernet frame. The frame should be a `frags_t` containing a complete Ethernet frame, excluding the preamble and SFD, but including space for the CRC field. The passed *frame* must not be modified by the called function.

The *replace_crc* flag indicates whether the CRC field contents can be modified by the implementing object: if *replace_crc* is not set, the implementing object will leave the CRC field untouched; if *replace_crc* is set, the implementing object is free to rewrite the CRC field according to the link constraints. Note that in many cases, setting *replace_crc* to true will allow the link to assume the CRC field to be matching the frame contents, thus skipping CRC calculation and improving simulation performance. *replace_crc* should only be set to false when the device wants to send a frame with a CRC field not matching the frame contents.

If the function return 0, the frame was sent to the link; In case -1 is returned, there was not enough bandwidth available right now, and the frame could not be sent. The PHY should call the *tx_bandwidth_available* in the `ieee_802_3_mac_v3` interface at the MAC, when the frame can be sent.

The *check_tx_bandwidth* can also be used to check that there is bandwidth available, without sending a frame. It returns 0 if there is no bandwidth available, and a positive value if the frame can be sent right away.

Execution Context

Instruction Context for all methods.

image

Implemented By

[image](#)

Description

This interface is used for handling big data images.

read() and *write* access a chunk of data at a time. Only accesses within the bounds of the image are allowed.

clear_range() fills an interval with null bytes, *fill()* with any byte value.

size() returns the image size.

get() and *set()* work like *read()* and *write()* but pass the data using a *bytes_t* instead, and can be used from Python.

Other methods are not currently for public use.

```
SIM_INTERFACE(image) {
#ifndef !defined(GULP)
    void (*read)(conf_object_t *img, void *NOTNULL to_buf, uint64 start,
                size_t length);
    void (*write)(conf_object_t *img, const void *NOTNULL from_buf,
                  uint64 start, size_t length);
    int (*for_all_spages)(conf_object_t *img,
                          int (*NOTNULL f)(image_spage_t *NOTNULL p,
                                           uint64 ofs, void *arg),
                          void *arg);
#endif /* not GULP */
    void (*set_persistent)(conf_object_t *obj);
    void (*save_to_file)(conf_object_t *NOTNULL obj,
                         const char *NOTNULL file,
                         uint64 start, uint64 length);
    void (*save_diff)(conf_object_t *NOTNULL obj,
                      const char *NOTNULL file);
    void (*clear_range)(conf_object_t *NOTNULL obj,
                        uint64 start, uint64 length);
    void (*fill)(conf_object_t *NOTNULL obj,
                 uint64 start, uint64 length, uint8 value);
    void (*dealloc_icode_page)(conf_object_t *NOTNULL obj,
                               image_spage_t *NOTNULL spage);
    uint64 (*size)(conf_object_t *NOTNULL obj);
    void (*set)(conf_object_t *NOTNULL obj, uint64 ofs, bytes_t b);
    bytes_t (*get)(conf_object_t *NOTNULL obj, uint64 ofs, size_t size);
};
```

Execution Context

read Instruction Context

<i>write</i>	Instruction Context
<i>for_all_spages</i>	Instruction Context
<i>set_persistent</i>	Instruction Context
<i>save_to_file</i>	Execution Context
<i>save_diff</i>	Execution Context
<i>clear_range</i>	Instruction Context
<i>dealloc_icode_page</i>	Execution Context
<i>size</i>	Execution Context
<i>fill</i>	Execution Context
<i>get</i>	Execution Context
<i>set</i>	Execution Context

int_register

Implemented By

[AM79C973](#), [DEC21140A-dml](#), [NS16450](#), [NS16550](#), [PMC1553](#), [TSB12LV26](#), [dm9161](#), [firewire_sample_device](#), [generic_eth_phy](#), [generic_PCIE_Switch_Port](#), [generic_SPI_Flash](#), [i21154](#), [i82559](#), [onfi_flash](#), [rapidio_simple_device](#), [tsi500](#)

Description

The `int_register` interface is used for access to registers in a processor. It can be used to access any kind of integer register, not only the “normal” registers. This includes all kinds of control registers, hidden registers and anything else that might be useful to access as a register. The only limitation is that the register value should be representable as an `uinteger_t` (a 64-bit unsigned integer).

This interface can be implemented by other classes than processors, but it is likely to be found mostly in processors.

Registers are identified by a number, and there are two functions to translate from register names to register numbers and back. The translation need not be one-to-one, which means that one register can have several names. A register name can, however, only translate to a single register number.

Often, registers are grouped in *register banks*, where registers in the bank are numbered from 0 up. Registers in a bank should have consecutive numbers (unless their numbering is very sparse). This allows a user to deduce register numbers by calling `get_number` for the first register only. The first register numbers should be used for the general-purpose integer registers, if possible (so that integer register `rN` has number `N`).

Using this interface to read or write registers does not cause any side effects, such as triggering interrupts or signalling haps.

`get_number` translates a register name to its number. A user defined `int_register` interface should return -1 if the register does not exist. Instances of this interface implemented by Simics itself may currently raise a frontend exception on errors.

`get_name` translates a register number to its canonical name.

`read` reads a register value.

`write` writes a new register value.

`all_registers` returns a list of all register numbers that can be used for this object.

`register_info` returns information about a single register. The information return depends on the `info` parameter.

Sim_RegInfo_Catchable

Return 1 if `Core_Control_Register_Write` and `Core_Control_Register_Read` are triggered when this register is written or read. Return 0 otherwise.

```
typedef enum {
    Sim_RegInfo_Catchable
} ireg_info_t;
```

```

SIM_INTERFACE(int_register) {
    int (*get_number)(conf_object_t *NOTNULL obj,
                      const char *NOTNULL name);
    const char *(*get_name)(conf_object_t *NOTNULL obj, int reg);
    uinteger_t (*read)(conf_object_t *NOTNULL obj, int reg);
    void (*write)(conf_object_t *NOTNULL obj, int reg, uinteger_t val);
    attr_value_t (*all_registers)(conf_object_t *NOTNULL obj);
    int (*register_info)(conf_object_t *NOTNULL obj, int reg,
                         ireg_info_t info);
};

#define INT_REGISTER_INTERFACE "int_register"

```

Execution Context

Instruction Context for all methods, except for *write* where the register is a program counter; Execution Context in that case.

Command List

<code>break-cr</code>	break on control register updates
<code>read-reg</code>	read a register
<code>register-number</code>	<i>deprecated</i> — get the number of a processor register
<code>trace-cr</code>	trace control register updates
<code>unbreak-cr</code>	break on control register updates
<code>untrace-cr</code>	trace control register updates
<code>wait-for-register-read</code>	wait for a register read
<code>wait-for-register-write</code>	wait for a register write
<code>write-reg</code>	write to register

Command Descriptions

<int_register>.break-cr

Synopsis

```

<int_register>.break-cr ("register"-|all|-list)
<int_register>.unbreak-cr ("register"-|all|-list)
break-cr ("register"-|all|-list)
unbreak-cr ("register"-|all|-list)

```

Description

Enables and disables breaking simulation on control register updates. When this is enabled, every time the specified control register is updated during simulation a message is printed. The message will name the register being updated, and the new value. The new value will be printed even if it is identical to the previous value.

The *reg-name* parameter specifies which control register should be traced. The available control registers depends on the simulated target.

Instead of a register name, the `-all` flag may be given. This will enable or disable tracing of all control register.

Using the `-list` argument will print out the registers accesses on which a breakpoint will trigger.

The non-namespace variant of this command breaks the simulation on control register updates on all processors of the same class in the same group as the currently inspected processor. The namespace variant affects only the selected processor.

See Also

[trace-cr](#), [<breakpoint>.break](#), [log-setup](#)

<int_register>.read-reg

Synopsis

```
<int_register>.read-reg "reg-name"  
read-reg [cpu-name] "reg-name"
```

Description

This command reads a register from an object implementing the `int_register` interface. For example, to read the `eax` register in an x86 processor called `cpu0`, write `read-reg cpu0 eax`. You can also use the method variant: `cpu0.read-reg eax`, or the more convenient variant `%eax` that reads a register from the selected frontend CPU.

If no `cpu-name` is supplied, the current frontend processor is used.

See Also

[%](#), [write-reg](#), [pregs](#), [pselect](#)

<int_register>.register-number — *deprecated*

Synopsis

```
<int_register>.register-number "reg-name"
```

Description

This command is deprecated; use `<int_register>.wait-for-register-read` or `<int_register>.wait-for-register-write` instead.

Returns the register number for a named processor register. The register number is used as hap indexing for example.

See Also

[%](#), [read-reg](#)

<int_register>.trace-cr

Synopsis

```
<int_register>.trace-cr ("register"-all|-list)
<int_register>.untrace-cr ("register"-all|-list)
trace-cr ("register"-all|-list)
untrace-cr ("register"-all|-list)
```

Description

Enables and disables tracing of control register updates. When this is enabled, every time the specified control register is updated during simulation a message is printed. The message will name the register being updated, and the new value. The new value will be printed even if it is identical to the previous value.

The *reg-name* parameter specifies which control register should be traced. The available control registers depends on the simulated target.

Instead of a register name, the `-all` flag may be given. This will enable or disable tracing of all control register.

Using the `-list` argument will print out the registers accesses currently being traced. The non-namespace variant of this command traces control register updates on all processors of the same class in the same group as the currently inspected processor. The namespace variant affects only the selected processor.

See Also

[break-cr](#), [log-setup](#)

<int_register>.unbreak-cr

Synopsis

```
<int_register>.unbreak-cr ("register"-all|-list)
<int_register>.break-cr ("register"-all|-list)
break-cr ("register"-all|-list)
unbreak-cr ("register"-all|-list)
```

Description

Enables and disables breaking simulation on control register updates. When this is enabled, every time the specified control register is updated during simulation a message is printed. The message will name the register being updated, and the new value. The new value will be printed even if it is identical to the previous value.

The *reg-name* parameter specifies which control register should be traced. The available control registers depends on the simulated target.

Instead of a register name, the `-all` flag may be given. This will enable or disable tracing of all control register.

Using the `-list` argument will print out the registers accesses on which a breakpoint will trigger.

The non-namespace variant of this command breaks the simulation on control register updates on all processors of the same class in the same group as the currently inspected processor. The namespace variant affects only the selected processor.

See Also

[trace-cr](#), [<breakpoint>.break](#), [log-setup](#)

<int_register>.untrace-cr

Synopsis

```
<int_register>.untrace-cr ("register"|-all|-list)
<int_register>.trace-cr ("register"|-all|-list)
trace-cr ("register"|-all|-list)
untrace-cr ("register"|-all|-list)
```

Description

Enables and disables tracing of control register updates. When this is enabled, every time the specified control register is updated during simulation a message is printed. The message will name the register being updated, and the new value. The new value will be printed even if it is identical to the previous value.

The *reg-name* parameter specifies which control register should be traced. The available control registers depends on the simulated target.

Instead of a register name, the *-all* flag may be given. This will enable or disable tracing of all control register.

Using the *-list* argument will print out the registers accesses currently being traced.

The non-namespace variant of this command traces control register updates on all processors of the same class in the same group as the currently inspected processor. The namespace variant affects only the selected processor.

See Also

[break-cr](#), [log-setup](#)

<int_register>.wait-for-register-read

Synopsis

```
<int_register>.wait-for-register-read [-always] "reg-name"
```

Description

Postpones execution of a script branch until the *reg-name* register is read by the target software. Only registers that are “catchable”, i.e. that can trigger control register haps on accesses can be waited on. The command will fail if a non-catchable register is specified. The *-always* flag is internal and should not be used.

See Also

[script-branch](#), [<int_register>.wait-for-register-write](#)

<int_register>.wait-for-register-write

Synopsis

```
<int_register>.wait-for-register-write [-always] "reg-name"
```

Description

Postpones execution of a script branch until the *reg-name* register is written by the target software. Only registers that are “catchable”, i.e. that can trigger control register traps on accesses can be waited on. The command will fail if a non-catchable register is specified. The *-always* flag is internal and should not be used.

See Also

[script-branch](#), [<int_register>.wait-for-register-read](#)

<int_register>.write-reg

Synopsis

```
<int_register>.write-reg "reg-name" value
write-reg [cpu-name] "reg-name" value
```

Description

Use this command to set the value of a register on an object implementing the `int_register` interface. For example, to set the `eax` register on the x86 processor `cpu0` to 3, write `write-reg cpu0 eax 3`. You can also use the method variant: `cpu0.write-reg eax 3`.

This function may or may not have the correct side-effects, depending on target and register. If no *cpu-name* is given, the current frontend processor is used.

See Also

[%](#), [read-reg](#), [pregs](#), [pselect](#)

io_memory

Implemented By

AM79C973, BCM5703C, BCM5704C, CL-PD6729, DEC21041, DEC21140A, DEC21140A-dml, DEC21143, ISP1040, ISP2200, NS16450, NS16550, NS16550_c, PMC1553, SIL680A, SYM53C810, SYM53C875, TSB12LV26, accel-vga, empty-device-c, empty-device-cc, fake-space, firewire_bus, firewire_sample_device, flash-memory, generic_pcie_switch_port, generic_spi_flash, hostfs, i21150, i21152, i21154, i21554-prim, i21554-scnd, i21555-prim, i21555-scnd, i82543, i82546, i82559, ide, onfi_flash, pci-bus, pcie-bus, rapidio_simple_device, set-memory, vga_pci, vmcom

Description

This interface is implemented by devices that can be mapped into address spaces (including port spaces). The *map()* function is called for an object when it initially is mapped into the address space, and *operation()* is called when the object is accessed through an address space.

The *obj* argument is a pointer to the mapped object and *map_info* contains information about how and where the device is mapped into memory. The *memory_or_io* argument to *map()* identifies the type of the address space where the device is mapped. The *mem_op* argument to *operate()* contains information about the access.

The offset into the device mapping for the access is typically calculated in the following way:

```
offset = mem_op->physical_address - map_info.base + map_info.start
```

The return value from *map()* is currently not used and should be always be 0.

The *exception_type_t* type, that is the same as *pseudo_exception_t*, returned by the *operation()* function may be used to signal errors to Simics, but should in most cases be *Sim_PE_No_Exception*. If the device does not support inquiry accesses, it should return *Sim_PE_Inquiry_Unhandled* if *mem_op->inquiry* is 1.

```
typedef enum {
    Sim_Addr_Space_Conf,
    Sim_Addr_Space_IO,
    Sim_Addr_Space_Memory
} addr_space_t;

SIM_INTERFACE(io_memory) {
    int (*map)(conf_object_t *NOTNULL obj,
               addr_space_t memory_or_io, map_info_t map_info);
    exception_type_t (*operation)(conf_object_t *NOTNULL obj,
                                  generic_transaction_t *NOTNULL mem_op,
                                  map_info_t map_info);
};
```

```
#define IO_MEMORY_INTERFACE "io_memory"
```

Execution Context

Instruction Context for all methods.

keyboard

Implemented By

[usb_hs_keyboard](#), [usb_keyboard](#)

Description

Interface implemented by keyboard controllers. Used by consoles to send keyboard events to the controller.

The function *keyboard_event()* takes the keyboard controller as its first argument *obj*. The *key_up* argument specifies whether the event is a key release (1) or a key press (0). The *key* argument is the Simics internal keycode, as defined in `src/extension/keycode-common/keycodes.h`.

If the return value is 1 the keyboard controller accepted the event. If return value is 0 the keyboard controller did not accept the event, and the console should buffer the event until it gets a *keyboard_ready()* call from the keyboard controller.

```
SIM_INTERFACE(keyboard) {
    int (*keyboard_event)(conf_object_t *obj, int key_up, uint8 key);
};

#define KEYBOARD_INTERFACE "keyboard"
```

Execution Context

Instruction Context for all methods.

log_object

Implemented By

AM79C973, AT24Cxx, BCM5703C, BCM5704C, CL-PD6729, DEC21041, DEC21140A, DEC21140A-dml, DEC21143, ISP1040, ISP2200, NS16450, NS16550, NS16550_c, PCF8582C, PMC1553, SIL680A, SYM53C810, SYM53C875, TSB12LV26, accel-vga, attr-meter, base-trace-mem-hier, bitmask-translator, branch_recorder, breakpoints, cell, cell-and-clocks, central-client, central-server, cli, clipboard-gateway, clock, component, connector, context, coverage_profiler, cp3_quad100tx, cpu-group, data-profiler, datagram_link, datagram_link_endpoint, datagram_link_impl, ddr-memory-module, ddr2-memory-module, ddr3-memory-module, disassemble_v9, disassemble_x86, dm9161, dummy-component, dynamic_link_connector, empty-device-c, empty-device-cc, etg, etg_comp, etg_panel, eth-cable-link, eth-cable-link-endpoint, eth-hub-link, eth-hub-link-endpoint, eth-link-snoop-endpoint, eth-probe, eth-probe-port, eth-switch-link, eth-switch-link-endpoint, eth-switch-link-snoop-endpoint, eth-transceiver, eth_injector, eth_injector_comp, ethernet-link, ethernet_cable, ethernet_hub, ethernet_switch, ethernet_vlan_switch, example-keypad, example-status-panel, fake-space, fc-disk, file-cdrom, firewire_bus, firewire_device_test_wrapper, firewire_sample_device, flash-memory, frequency_bus, frontend-server, frontend-server-console, ftp-alg, ftp-control, ftp-data, ftp-service, g-cache, gdb-remote, generic-flash-memory, generic-message-link, generic-message-sample-device, generic mmc-card, generic_eth_phy, generic_pcie_switch, generic_pcie_switch_port, generic_spi_flash, gfx-console, gui, hap-meter, host-condor-pci-1553, host-pmc-1553, host-serial-console, hostfs, hypersim-pattern-matcher, i21150, i21152, i21154, i21554-prim, i21554-scnd, i21555-prim, i21555-scnd, i2c-bus, i2c-link-endpoint, i2c-link-impl, i2c_link_v1, i2c_link_v2, i2c_slave_v2_to_bus_adapter, i2c_wire, i82543, i82546, i82559, id-splitter, ide, ide-cdrom, ide-disk, image, instruction-data-splitter, isa-fourport, isa-lance, isa-vga, mac_splitter, mem-traffic-meter, memory-space, memory-timer, micron_mtfc2ggqdi_emmc_card, micron_mtfc4ggqdi_emmc_card, micron_mtfc4ggqdi_sdhc_card, microwire-eeprom, mii-management-bus, mii-transceiver, ms1553-link, ms1553-recorder-bm, ms1553-test-rt, ms1553_rt, mtprof, onfi_flash, os Awareness, pc-dual-serial-ports, pc-floppy-controller, pc-quad-serial-ports, pc-single-parallel-port, pci-accel-vga, pci-am79c973, pci-bcm5703c, pci-bcm5704c, pci-bus, pci-dec21041, pci-dec21140a, pci-dec21140a-dml, pci-dec21143, pci-i21152, pci-i82543gc, pci-i82546bg, pci-i82559, pci-ispl040, pci-ispl2200, pci-pd6729, pci-pmc1553, pci-sil680a, pci-sym53c810, pci-sym53c875, pci-sym53c876, pci-tsb12lv26, pci-vga, pcie-bus, perfanalyze-client, persistent-ram, phy-mii-transceiver, phy_comp, port-forward-incoming-server, port-forward-outgoing-server, port-space, preferences, ps2-keyboard-mouse, ram, rapidio_link, rapidio_link_endpoint, rapidio_link_impl, rapidio_simple_device, rapidio_tape, real-network-bridge, real-network-host, real-network-router, realtime_recorder, remote_sync_domain, remote_sync_node, remote_sync_server, rev-execution, rn-eth-bridge-raw, rn-eth-bridge-tap, rn-eth-proxy-raw, rn-ip-router-raw, rom, sample-gcache, scsi-bus, scsi-cdrom, scsi-disk, sdram-memory-module, selfprof, ser-link-endpoint, ser-link-impl, ser_link, serial-link, service-node, service-node-device, set-memory, signal-bus, signal_link, signal_link_endpoint, signal_link_impl, signal_to_interrupt, sim, simple-fc-disk, simple-scsi-disk, simple_memory_module, sio-w83627hf, slaver, software_tracker, state-assertion, static_link_connector, status_panel, status_

panel_view, std-etg, std-ethernet-link, std-firewire-bus, std-firewire-sample-device, std-generic-link, std-generic-link-sample, std-graphics-console, std-host-serial-console, std-ide-cdrom, std-ide-disk, std-ms1553-link, std-pcmcia-flash-disk, std-scsi-bus, std-scsi-cdrom, std-scsi-disk, std-serial-link, std-server-console, std-service-node, std-superio, std-telnet-console, std-text-console, std-text-graphics-console, std_mmc_card, std_sata_cdrom, std_sata_disk, symtable, sync_domain, system_panel_bool_in, system_panel_bool_out, system_panel_number_in, system_panel_number_out, system_panel_state_manager, tcf-agent, tcf-context-proxy, telnet_console, telnet_frontend, terminal_frontend, text-console, text_panel_frontend, time-server, top-component, trace-memhier, trace-sync, trans-sorter, trans-splitter, trans-staller, tsi500, usb-disk, usb-santa, usb-tablet-comp, usb-wacom-tablet-comp, usb_disk, usb_hs_keyboard, usb_hs_keyboard_comp, usb_keyboard, usb_keyboard_comp, usb_mouse, usb_mouse_comp, usb_santa, usb_tablet, usb_wacom_tablet, usb_xmas_tree, vga_pci, vmcom, wdb-remote

Description

Internal.

Execution Context

Not applicable.

map_demap

Implemented By

[memory-space](#), [port-space](#)

Description

Interface used to dynamically add and remove mappings in a memory space. The first argument to all functions, *space* is the memory space to map an object in.

The *map_simple* function adds a device/port pair to the memory map. The added device (port) must implement either the *_ram*, *_rom*, *io_memory* or *port_space*, or it can be another memory space (which implements the *translate* and *memory_space* interfaces).

The *map_bridge* function adds a device/port pair to the memory map to act as a bridge that translates memory accesses before passing them on to another object. The added device (port) must implement either the *translate* interface or the *bridge* interface, and the *target* and *target_port* parameter specifies the object behind the bridge. The target is typically another memory space.

The *unmap* removes an object from the memory map. The *dev* and *port* parameters are the same as those given to the *map_simple* and *map_bridge* functions.

The *unmap_address* removes the object from the memory map that is mapped at the address specified by the *base* parameter.

The *add_map*, *remove_map*, *add_default* and *remove_default* functions are deprecated and should not be used.

All functions in the *map_demap* return 1 on success and 0 on failure.

More information about the different kinds of memory space mappings is available in Simics Model Builder User Guide.

```
SIM_INTERFACE(map_demap) {
    /* old-style */
    int (*add_map)(conf_object_t *NOTNULL space,
                   conf_object_t *NOTNULL dev,
                   conf_object_t *target,
                   map_info_t map_info);
    int (*remove_map)(conf_object_t *NOTNULL space,
                      conf_object_t *NOTNULL dev,
                      int function);
    int (*add_default)(conf_object_t *NOTNULL space,
                       conf_object_t *NOTNULL dev,
                       conf_object_t *target,
                       map_info_t map_info);
    void (*remove_default)(conf_object_t *NOTNULL space);

    /* new-style */
    int (*map_simple)(conf_object_t *NOTNULL space,
```

```

        conf_object_t *NOTNULL dev,
        const char *dev_port,
        map_info_t map_info);
int (*map_bridge)(conf_object_t *NOTNULL space,
                  conf_object_t *NOTNULL dev,
                  const char *dev_port,
                  conf_object_t *target,
                  const char *target_port,
                  map_info_t map_info);
int (*unmap)(conf_object_t *NOTNULL space,
             conf_object_t *NOTNULL dev,
             const char *dev_port);
int (*unmap_address)(conf_object_t *NOTNULL space,
                     conf_object_t *NOTNULL dev,
                     physical_address_t base,
                     const char *dev_port);
};

#define MAP_DEMAP_INTERFACE "map_demap"

```

Execution Context

Execution Context for all methods.

mdio45_bus

Implemented By
[mii-management-bus](#)

Description

```
SIM_INTERFACE(mdio45_bus) {
    uint16 (*read_register)(conf_object_t *obj, int phy, int mmd, int reg);
    void   (*write_register)(conf_object_t *obj, int phy, int mdd, int reg,
                           uint16 value);

};

#define MDIO45_BUS_INTERFACE "mdio45_bus"
```

Interface that should be implemented by classes that either represent an MDIO bus, or multi PHY devices implementing the IEEE 802.3 clause 45 MDIO management interface.

The *read_register* function should return the 16-bit value of the specified register. *phy* specifies the PHY address (0-31), *mmd* specifies the MMD (MDIO manageable device) within the PHY (0-31) and *reg* specifies the register number within the MMD (0-65535).

Devices modeling a single PHY should implement the mdio45_phy interface instead.

Execution Context

Instruction Context for all methods.

mdio45_phy

Implemented By
[generic_eth_phy](#)

Description

```
SIM_INTERFACE(mdio45_phy) {
    uint16 (*read_register)(conf_object_t *obj, int mmd, int reg);
    void   (*write_register)(conf_object_t *obj, int mmd, int reg,
                           uint16 value);

};

#define MDIO45_PHY_INTERFACE "mdio45_phy"
```

Interface that should be implemented by classes representing a single PHY implementing the IEEE 802.3 clause 45 MDIO management interface.

The *read_register* function should return the 16-bit value of the specified register. *mmd* specifies the MMD (MDIO manageable device) within the PHY (0-31) and *reg* specifies the register number within the MMD (0-65535).

Devices modeling either an MDIO bus, or multiple PHYs should implement the *mdio45_bus* interface instead.

Execution Context

Instruction Context for all methods.

memory_space

Implemented By

[fake-space](#), [memory-space](#)

Description

This interface is implemented by classes that provide linear address spaces. Other objects may perform accesses in the address space using the *access()* function or one of the simplified access functions, or may ask for mapped objects using *space_lookup()*. Typical usage of this interface would be memory accesses from devices or processors.

The *access_simple()* function is similar to *access()* but takes some additional arguments instead of a complete *generic_transaction_t* structure. An inquiry version of this function is available as *access_simple_inq()*. Both these functions can perform endian conversion if the buffer pointed to by *buf* contains a value in host endian byte-order. To avoid endian conversion, *Sim_Endian_Target* should be specified as *endian*. These two functions are not available from Python.

The *read()* and *write()* methods are primarily intended for use in Python. They operate on data encapsulated in attribute values which should be of data type; in Python, this corresponds to tuples of integers (bytes). These two methods signal errors by raising a frontend exception. They can be used to read or write up to 1024 bytes in a single call. The return value from *write()* should be ignored.

The *fill()* method fills *size* bytes starting at *start* with *value*. It only works on memory, and returns the number of bytes actually written before encountering something that isn't RAM.

```
SIM_INTERFACE(memory_space) {
    map_list_t * (*space_lookup)(conf_object_t *NOTNULL obj,
                                generic_transaction_t *NOTNULL mop,
                                map_info_t mapinfo);
    exception_type_t (*access)(conf_object_t *NOTNULL obj,
                               generic_transaction_t *NOTNULL mop);
#if !defined(GULP)
    exception_type_t (*access_simple)(conf_object_t *obj,
                                      conf_object_t *initiator,
                                      physical_address_t addr,
                                      uint8 *buf,
                                      physical_address_t len,
                                      read_or_write_t type,
                                      endianness_t endian);
    exception_type_t (*access_simple_inq)(conf_object_t *obj,
                                         conf_object_t *initiator,
                                         physical_address_t addr,
                                         uint8 *buf,
                                         physical_address_t len,
                                         read_or_write_t type,
                                         
```

```

                                endianness_t endian);

#endif /* !GULP */
attr_value_t (*read)(conf_object_t *NOTNULL obj,
                     conf_object_t *initiator,
                     physical_address_t addr,
                     int length,
                     int inquiry);
exception_type_t (*write)(conf_object_t *NOTNULL obj,
                          conf_object_t *initiator,
                          physical_address_t addr,
                          attr_value_t data,
                          int inquiry);
cycles_t (*timing_model_operate)(conf_object_t *NOTNULL space,
                                  generic_transaction_t *NOTNULL mop);
uint64 (*fill)(conf_object_t *NOTNULL obj,
                physical_address_t start, uint64 size, uint8 value,
                bool inquiry);
};

#define MEMORY_SPACE_INTERFACE "memory_space"

```

Execution Context

Intruction Context for all methods.

Command List

[debug](#) get debug object

Command Descriptions

<memory_space>.debug

Synopsis

<memory_space>.debug ["name"]

Description

Get a debug context for the memory space or processor.

microwire

Implemented By
[microwire-eeprom](#)

Description

This interface is used to communicate with a 3-wire (microwire) serial EEPROM device, for example, the 93Cxx series, via its pins. To set the values of the CS, SK, and DI pins, use the *set_cs()*, *set_sk()*, and *set_di()* functions, respectively. To read the value output on the DO pin, use the *get_do()* function. Zero represents low signal and non-zero high signal. The *read_word()* and *write_word()* functions are provided to shortcut high-level operations. They operate on 8-bit or 16-bit entities depending on the width of the eeprom.

```
#define MICROWIRE_INTERFACE "microwire"

SIM_INTERFACE(microwire) {
    void (*set_cs)(conf_object_t *obj, int cs);
    void (*set_sk)(conf_object_t *obj, int sk);
    void (*set_di)(conf_object_t *obj, int di);
    int  (*get_do)(conf_object_t *obj);
    uint16 (*read_word)(conf_object_t *obj, int offset);
    void  (*write_word)(conf_object_t *obj, int offset, uint16 value);
};
```

Execution Context

Instruction Context for all methods.

mii

Implemented By

[dm9161](#), [eth-transceiver](#), [generic_eth_phy](#), [mii-transceiver](#)

Description

```
SIM_INTERFACE(mii) {
    int      (*serial_access)(conf_object_t *obj, int data_in, int clock);
    uint16   (*read_register)(conf_object_t *obj, int index);
    void     (*write_register)(conf_object_t *obj, int index, uint16 value);
};

#define MII_INTERFACE "mii"
```

Obsolete interface that is implemented by some PHY's that support the MII management interface.

It has the same methods as the mii-management interface, but does not pass along the PHY number.

To support low-level bitwise accesses via MDIO and MDC pins, the function *serial_access* can be used. It is recommended to leave this function unimplemented and let an instance of **mii-management-bus** convert the low-level bit operations to higher level read/write register calls.

Execution Context

Instruction Context for all methods.

mii_management

Implemented By

[dm9161](#), [eth-transceiver](#), [generic_eth_phy](#), [mii-management-bus](#), [mii-transceiver](#)

Description

```
SIM_INTERFACE(mii_management) {
    int      (*serial_access)(conf_object_t *obj, int data_in, int clock);
    uint16   (*read_register)(conf_object_t *obj, int phy, int reg);
    void     (*write_register)(conf_object_t *obj, int phy, int reg,
                               uint16 value);
};

#define MII_MANAGEMENT_INTERFACE "mii_management"
```

Interface that should be implemented by classes that represents one or multiple PHY's that have MII management interfaces.

The *read_register* function should return the 16-bit value of the specified register. There are 32 registers numbered 0-31. The phy argument indicates the PHY number (0-31). Classes that represents one PHY can ignore this argument. The *write_register* function is called when a register is written.

To support low-level bitwise accesses via MDIO and MDC pins, the function *serial_access* can be used. It is recommended to leave this function unimplemented and let an instance of **mii-management-bus** convert the low-level bit operations to higher level read/write register calls. The *serial_access* function takes as argument the MDIO and MDC pin levels on the master, and return the MDIO pin from the slave. Note that **mii-management-bus** also have signal interfaces for these pins.

Execution Context

Instruction Context for all methods.

mmc

Implemented By
[generic mmc card](#)

Description

```
#define MMC_INTERFACE "mmc"
SIM_INTERFACE(mmc) {
#ifndef !defined(GULP)
    int (*send_command)(conf_object_t *obj, uint8 cmd, uint32 args,
                        buffer_t response);
    int (*read_data)(conf_object_t *obj, buffer_t data);
#endif
    int (*write_data)(conf_object_t *obj, bytes_t data);
};
```

Interface that should be implemented by all MMC/SD/SDHC/SDIO card models.

send_command: sends a 5-byte command to the card (1-byte command index and 4 bytes command arguments). Caller provides the response length. Card fills in actual response data. The response data is 0, 6 or 17 bytes, in big-endian (see the MMC/SD specification for details). Return value: number of response bytes, -1 if the command wasn't accepted (e.g. command is not supported or illegal in current state, or command is not supported or illegal for current card type).

read_data: reads data. Caller provides the length. Return value: the card fills in the provided buffer, and returns the number of bytes actually read, which might be less than the buffer length in case of error.

write_data: writes data. Caller provides in both length and data. Return value: number of bytes actually written, which might be less than the provided data length in case of error.

Execution Context

Instruction Context for all methods.

ms1553_bridge_terminal

Implemented By
[ms1553_rt](#)

Description

```
SIM_INTERFACE(ms1553_bridge_terminal) {
    void (*new_message)(conf_object_t *obj, conf_object_t *bridge,
                        uint16 cmd, ms1553_words_t data, uint16 sts);
};

#define MS1553_BRIDGE_TERMINAL_INTERFACE "ms1553_bridge_terminal"
```

This interface should be implemented by MIL-STD-1553 remote terminals in Simics connected via a bridge to a host 1553 card which is connected to a real MIL-STD-1553 bus. The interface is described in the documentation for the `ms1553_bridge_bus` interface.

Execution Context

Instruction Context for all methods.

ms1553_link

Implemented By

[host-pmc-1553](#), [ms1553-link](#)

Description

```
SIM_INTERFACE(ms1553_link) {
    int (*connect_terminal)(conf_object_t *link,
                           conf_object_t *NOTNULL obj);
    void (*disconnect_terminal)(conf_object_t *link, int id);
    void (*send_data)(conf_object_t *link, int id,
                      ms1553_phase_t phase, ms1553_words_t data);
    void (*set_terminal_address)(conf_object_t *link, int id, int address);
    void (*clr_terminal_address)(conf_object_t *link, int id);
    void (*set_idle)(conf_object_t *link, int id);
    void (*inject_data)(conf_object_t *link,
                        ms1553_phase_t phase, ms1553_words_t data);
    void (*inject_error)(conf_object_t *link,
                         ms1553_error_t err);
};

#define MS1553_LINK_INTERFACE "ms1553_link"
```

This interface is implemented by the MIL-STD-1553 link objects that provide a protocol level interface for message delivery.

The *connect_terminal()* function attaches an 1553 device to the link. The device must implement the `ms1553_terminal` interface. The return value is an identification number that should be used in subsequent calls to the link to identify the device. On failure, for example if there is not room for more terminals on the link, the connect function will return -1. Terminals should always reconnect to the link when starting from a checkpoint.

The *disconnect_terminal()* function detaches an 1553 device from the link. It will not receive any more data from the link and may not call any functions in the interface again, except *connect_terminal()*.

A remote terminal may, and should preferably, register its terminal address using the *set_terminal_address()* to only receive data destined for it. If no address is registered it will receive all data sent on the bus. The *clr_terminal_address()* is used to unregister a previously installed terminal address from the 1553 link. Models of bus monitors should typically not register any address, but instead receive all data traffic.

The *send_data()* function is used by all devices to send data words to the link. Devices must observe the 1553 protocol standards. I.e. a message starts with a command from the bus controller device. This command will be sent to the addressed remote terminal, or all of them in case of broadcast, using the *receive_data()* function in the `ms1553_`

terminal interface. The addressed terminal responds with status and data, in case of a “Transmit” command, using the *send_data()* function. For a “Receive” command, the bus controller sends the data with *send_data()* and the addressed remote terminal finally sends the status word.

Commands, status and data are always sent in separate calls to *send_data()*, where the *phase* parameter describes what kind is sent. The *phase* is provided to simplify the implementation of the protocol handling in devices, it is possible to figure out the kind of the sent data by decoding the words sent on the link as in a real case. The values in the `ms1553_phase_t` type correspond to the *Idle*, *Transmit Command*, *Receive Command*, *Transmit Mode Command*, *Receive Mode Command*, *No-data Mode Command*, *Data* and *Status* phases.

Commands, status and data words should always be stored in host endian byte order, when packed in a `ms1553_words_t` structure.

A terminal may not respond by calling *send_data()* directly from its own *receive_data()*. A simple way to delay the reply transfer is to post it as a Simics event.

The *set_idle()* function should be used by the bus controller to restore the protocol checking engine in the link after a protocol error has occurred, for example after an RT timeout.

Common link errors are reported by the link using *receive_error()*, simplifying the implementation of device models. Some error types are not detected in Simics itself, such as Manchester decoding errors, due to the modeling abstraction in Simics but may be injected to test error handling in drivers.

Errors can also be injected in the link with *inject_error()*, and terminals should be prepared to handle calls to their *receive_error()*.

Data traffic can be also be injected with *inject_data()*, to simulate generic traffic without registering a device.

Execution Context

Instruction Context for all methods.

ms1553_terminal

Implemented By

[PMC1553](#), [ms1553-recorder-bm](#), [ms1553-test-rt](#), [ms1553_rt](#)

Description

```
SIM_INTERFACE(ms1553_terminal) {
    void (*receive_data)(conf_object_t *obj, conf_object_t *link, int id,
                         ms1553_phase_t phase, ms1553_words_t data);
    void (*receive_error)(conf_object_t *obj, conf_object_t *link, int id,
                          ms1553_error_t err);
};

#define MS1553_TERMINAL_INTERFACE "ms1553_terminal"
```

This interface should be implemented by MIL-STD-1553 devices, such as bus controllers and remote terminals. Its use is described in the documentation of the `ms1553_link` interface.

Execution Context

Instruction Context for all methods.

nand_flash

Implemented By

[onfi_flash](#)

Description

The nand_flash interface is used to perform read and write accesses and control some pins on NAND Flash devices.

The *read_access()* and *write_access()* functions perform read and write accesses.

The *set_command_latch_enable()*, *set_address_latch_enable()*, *set_write_protect()* and *set_spare_area_enable()* functions set the input level of the pins with the same names. 0 represents low input and 1 represents high input.

The chip enable and ready/busy pins are not modelled. The NAND Flash device assumes that chip enable is always asserted, and the device is never busy.

```
#define NAND_FLASH_INTERFACE "nand_flash"

SIM_INTERFACE(nand_flash) {
    uint16 (*read_access) (conf_object_t *obj);
    void (*write_access) (conf_object_t *obj, uint16 value);

    void (*set_command_latch_enable) (conf_object_t *obj, int value);
    void (*set_address_latch_enable) (conf_object_t *obj, int value);
    void (*set_write_protect) (conf_object_t *obj, int value);
    void (*set_spare_area_enable) (conf_object_t *obj, int value);
};
```

Execution Context

Instruction Context for all methods.

path_translate

Implemented By
[symtable](#)

Description

This interface provides path name translations.

The *lookup_readable* function takes a file path and returns the name of a existing, readable file based on the given name. No translation found a readable file, and the original file name wasn't readable, a nil value is returned.

The *translations* function returns a list of all possible translations of the given file path. This is the list of translations that the *lookup* function tries, in the same order, before it tries the unmodified file path. The returned list does not contain the original file path.

```
SIM_INTERFACE(path_translate) {
    /* Look up a file, and return a file name of an existing file, or
     * nil. */
    attr_value_t (*lookup_readable)(conf_object_t *obj,
                                   const char *NOTNULL path);

    /* Return all possible translations of path. These are the paths that
     * lookup try, in the same order, before it tries the unchanged
     * path. */
    attr_value_t (*translations)(conf_object_t *obj,
                                const char *NOTNULL path);
};

#define PATH_TRANSLATE_INTERFACE "path_translate"
```

Execution Context

Instruction Context for all methods.

pci_express

Implemented By

[generic_pcie_switch_port](#), [pcie-bus](#)

Description

This interface can be implemented by any PCI Express device, switch or endpoint. It is also implemented by the pci-bus, which will pass it on to the other end; e.g. if the endpoint sends a message, the pci-bus will pass it on to the bridge (downport), and if the downport sends it, it will be broadcasted to all devices on the bus.

src is the object sending the object. *type* is one of:

```
typedef enum {  
    /* INTx emulation */  
    PCIE_Msg Assert INTA      = 0x20,  
    PCIE_Msg Assert INTB      = 0x21,  
    PCIE_Msg Assert INTC      = 0x22,  
    PCIE_Msg Assert INTD      = 0x23,  
    PCIE_Msg Deassert INTA    = 0x24,  
    PCIE_Msg Deassert INTB    = 0x25,  
    PCIE_Msg Deassert INTC    = 0x26,  
    PCIE_Msg Deassert INTD    = 0x27,  
  
    /* Power Management */  
    PCIE_PM_Active_State_Nak  = 0x14,  
    PCIE_PM_PME                = 0x18,  
    PCIE_PM_Turn_Off            = 0x19,  
    PCIE_PM_PME_TO_Ack          = 0x1B,  
  
    /* Error Messages */  
    PCIE_ERR_COR                = 0x30,  
    PCIE_ERR_NONFATAL           = 0x31,  
    PCIE_ERR_FATAL               = 0x33,  
  
    /* Locked Transaction */  
    PCIE_Locked_Transaction     = 0x00,  
  
    /* Slot Power Limit */  
    PCIE_Set_Slot_Power_Limit   = 0x90,  
  
    /* Hot Plug Messages */  
    PCIE_HP_Power_Indicator_On  = 0x45,  
    PCIE_HP_Power_Indicator_Blink = 0x47,  
    PCIE_HP_Power_Indicator_Off  = 0x44,  
    PCIE_HP_Attention_Button_Pressed = 0x48,  
    PCIE_HP_Attention_Indicator_On = 0x41,
```

```

PCIE_HP_Attention_Indicator_Blink = 0x43,
PCIE_HP_Attention_Indicator_Off   = 0x40,

/* Data Link Layer (virtual) Messages

NOTE: these messages only exist on Simics simulator, as they are
normally part of the Data Link Layer which is below the level of
abstraction for Simics PCIe models

According to PCIe rev 2.0, when a target receives a message that it
does not recognize or support, except for the "Vendor Defined Type
1" message, it should treat the request as an "Unsupported Request"
and report it accordingly (see sec 2.3.1 for reference).

Hence models that comply with rev 2.0 must be updated to either
1) handle these messages or 2) ignore these messages.

Ideally we would like to use a new pcie_link interface to transmit
this information - see bug 17849 for more info. */
PCIE_DLL_Link_Down = -1,
PCIE_DLL_Link_Up   = -2,
}

pcie_message_type_t;

```

The contents of *payload* depends on *type*.

```

SIM_INTERFACE(pci_express) {
    int (*send_message)(conf_object_t *dst, conf_object_t *src,
                        pcie_message_type_t type, byte_string_t payload);
};

#define PCI_EXPRESS_INTERFACE "pci_express"

```

Execution Context

Instruction Context for all methods.

pci_express_hotplug

Implemented By

[generic_pcie_switch_port](#)

Description

This interface is intended for PCI Express switches that need to monitor the status of their downports.

presence_change is called from the pci-bus when a device is added or removed from the bus. *is_present* is set to 1 when the device is added, and 0 when it is removed.

inject_power_fault can be used to simulate a power fault on the downport. It is never called automatically.

press_attention_button can be called to simulate the attention button being pressed. The switch can respond to this by e.g. raising an interrupt and setting appropriate status bits. It is never called automatically.

set_mrl_state is similar to *attention_button_press* but controls the Manually operated Retention Latch. Set *locked* to 1 to simulate it being locked/closed, and 0 to simulate it being unlocked/opened.

Finally, *get_mrl_state* returns the state of the Manually operated Retention Latch.

```
SIM_INTERFACE(pci_express_hotplug) {
    /* This is sent when a device is added or removed from the bus. */
    void (*presence_change)(conf_object_t *dst, conf_object_t *NOTNULL src,
                           int is_present);
    void (*inject_power_fault)(conf_object_t *obj);
    void (*press_attention_button)(conf_object_t *obj);
    void (*set_mrl_state)(conf_object_t *obj, int locked);
    int   (*get_mrl_state)(conf_object_t *obj);
};

#define PCI_EXPRESS_HOTPLUG_INTERFACE "pci_express_hotplug"
```

Execution Context

Instruction Context for all methods.

port_space

Implemented By
[port-space](#)

Description

I/O port interface.

```
SIM_INTERFACE(port_space) {
    exception_type_t (*port_operation)(conf_object_t *NOTNULL pspace_obj,
                                         generic_transaction_t *NOTNULL mop,
                                         map_info_t map_info);
    attr_value_t (*read)(conf_object_t *NOTNULL obj,
                         conf_object_t *initiator,
                         physical_address_t addr,
                         int length,
                         int inquiry);
    exception_type_t (*write)(conf_object_t *NOTNULL obj,
                              conf_object_t *initiator,
                              physical_address_t addr,
                              attr_value_t data,
                              int inquiry);
};

#define PORT_SPACE_INTERFACE "port_space"
```

Execution Context

Instruction Context for all methods.

rapidio_v3

Implemented By

[rapidio_simple_device](#), [rapidio_tape](#), [tsi500](#)

Description

```
SIM_INTERFACE(rapidio_v3) {
    void (*memory_operation)(conf_object_t *obj, uint16 destination,
                            uint16 source, physical_address_t addr,
                            dbuffer_t *buf, rapidio_operation_t op);
    void (*doorbell)(conf_object_t *obj, uint16 destination,
                     uint16 source, uint16 data);
    void (*deliver_message)(conf_object_t *obj, uint16 destination,
                           uint16 source, uint16 mbox,
                           uint16 letter, dbuffer_t *data);
    uint32 (*read_register)(conf_object_t *obj, uint16 destination,
                           uint16 source, uint8 hopcount, int reg_no);
    void (*write_register)(conf_object_t *obj, uint16 destination,
                          uint16 source, uint8 hopcount,
                          int reg_no, uint32 value);
    void (*port_write)(conf_object_t *obj, uint16 target_id,
                      uint16 source_id, uint8 hopcount, dbuffer_t *msg);
}

#define RAPIDIO_V3_INTERFACE "rapidio_v3"
```

The interface is designed for peer-to-peer communication. Switches and devices will use the same interface.

The *doorbell* method corresponds to a DOORBELL/DONE packet pair, while *deliver_message* denotes a non-fragmented message. On the physical line this will be split into 256-byte MESSAGE packets acknowledged by a DONE packet.

read_register and *write_register* is the way to simulate MAINTENANCE operations. I/O accesses are done using the *memory_operation* function, which will encode the operation in the *op* argument. The valid values are listed below.

```
typedef enum {
    RapidIO_Read,
    RapidIO_Write,
    RapidIO_Increment,
    RapidIO_Decrement,
    RapidIO_Test_and_Swap,
    RapidIO_Set,
    RapidIO_Clear,
    RapidIO_Compare_and_Swap,
```

```
    RapidIO_Swap  
} rapidio_operation_t;
```

The size of the *buf* argument decides the transaction size and the *addr* argument dictates where the access is made. The dbuffer *buf* serves as both incoming data and a place where the receiver can store read data. The bytes of a increment or decrement operation is in big endian order in the dbuffer.

port_write corresponds to PORT_WRITE packet, with all arguments stored in big endian order in the dbuffer argument. Typically this is the four 32-bit words component_tag_csr, port_error_detect_csr, port_id and transport_layer_error_detect_csr.

This interface has been superseded by `rapidio_v4`. Cache control packets like IO_READ_HOME and FLUSH are not supported yet. Error reporting is not supported.

Execution Context

Instruction Context for all methods.

rapidio_v5

Implemented By
[rapidio_link_endpoint](#)

Description

```
SIM_INTERFACE(rapidio_v5) {
    void (*transaction_request) (
        conf_object_t *obj, uint16 target_id, uint16 source_id,
        uint32 transport_class,
        uint64 addr, rapidio_operation_t op, bytes_t msg, uint64 id);
    void (*transaction_response) (
        conf_object_t *obj, bytes_t msg, uint64 id);
    void (*doorbell) (
        conf_object_t *obj, uint16 target_id, uint16 source_id,
        uint32 transport_class, uint16 val);
    void (*deliver_message) (
        conf_object_t *obj, uint16 target_id, uint16 source_id,
        uint32 transport_class,
        uint16 mbox, uint16 letter, bytes_t msg);
    void (*read_register_request) (
        conf_object_t *obj, uint16 target_id, uint16 source_id,
        uint32 transport_class,
        uint8 hopcount, uint32 reg_no, uint64 id);
    void (*read_register_response) (
        conf_object_t *obj, uint32 val, uint64 id);
    void (*write_register) (
        conf_object_t *obj, uint16 target_id, uint16 source_id,
        uint32 transport_class,
        uint8 hopcount, uint32 reg_no, uint32 val);
    void (*port_write) (
        conf_object_t *obj, uint16 target_id, uint16 source_id,
        uint32 transport_class,
        uint8 hopcount, bytes_t msg);
    void (*stream_data) (
        conf_object_t *obj, uint16 target_id, uint16 source_id,
        uint32 transport_class,
        uint8 cos, uint8 xh, uint16 stream_id, bytes_t pdu);
};

#define RAPIDIO_V5_INTERFACE "rapidio_v5"
```

This interface is designed for peer-to-peer RapidIO communication. It superseeds the `rapidio_v4` interface; adding data streaming support.

The request/response function pairs take an *id* parameter; the caller of the request function may choose any ID it likes, and the recipient is expected to reply by calling the sender's response function with the same ID. The recipient may delay for as long as it wishes before responding, but it is required to respond exactly once to each request. Both sides may continue to make new calls while a request is underway.

I/O accesses are done using *transaction_request* and *transaction_response*; the kind of access is determined by the *op* argument. The valid values are:

```
typedef enum {
    RapidIO_Read,
    RapidIO_Write,
    RapidIO_Increment,
    RapidIO_Decrement,
    RapidIO_Test_and_Swap,
    RapidIO_Set,
    RapidIO_Clear,
    RapidIO_Compare_and_Swap,
    RapidIO_Swap
} rapidio_operation_t;
```

The size of the *msg* argument decides the transaction size and the *addr* argument dictates where the access is made. The contents of *msg* is the input and output values of the transaction (assumed to be big-endian for the purposes of the increment and decrement operations). The plain read and write operations only send useful data in *msg* in one direction, but a byte sequence of the right length must be sent anyway (but its contents are ignored).

The *doorbell* method corresponds to a DOORBELL/DONE packet pair, while *deliver_message* denotes a non-fragmented message. On the physical line this will be split into 256-byte MESSAGE packets, acknowledged with a DONE packet.

read_register_request, *read_register_response*, and *write_register* is the way to simulate MAINTENANCE operations.

port_write corresponds to PORT_WRITE packet, with all arguments stored in big-endian order in the *msg* argument. Typically, this is the four 32-bit words *component_tag_csr*, *port_error_detect_csr*, *port_id*, and *transport_layer_error_detect_csr*.

The *stream_data* method corresponds to a non-fragmented DATA STREAM packet. On the physical line this will be split into MTU-sized MESSAGE packets.

From RapidIO specifications Part 10: "Data streaming transactions differ from most other RapidIO transactions in two ways: they must accommodate larger variably sized data transfers, and the transactions are not acknowledged with a response packet."

When *xh* is set to 1, the *stream_data* corresponds to the extended DATA STREAM packet used for traffic management. The payload for this extended packet is always four bytes, representing the TM_OP, wildcard, mask, parameter 1 and parameter 2 bits stored in big-endian order. Note that *xh* is the 3+1 bit pair representing Xtype and *xh* bits from rapidio packet header, not a bool.

The *transport_class* represents packet fields that are transmitted in the physical layer part. For serial RapidIO, bits are allocated as follows: critical request flow, `crf`, in bit 0, priority in bit 3-2, and virtual channel in bit 7-4.

This interface is not yet perfect, and may undergo non-compatible changes in the future. Error reporting and cache control packets like `IO_READ_HOME` and `FLUSH` are not yet supported.

Execution Context

Instruction Context for all methods.

recorder

Implemented By
[recorder](#)

Description

This interface is **deprecated**; please use the `recorder_v2` instead.

To initiate recording, the object should call the `attach` method with the recorder, the object itself, and an input handler as parameters. This is done once, typically when the object instance is created.

When the object receives data from outside the simulator, it should call the `input` method with a `dbufbuffer_t` containing the raw data in unspecified form, along with an extra word of user data. The `data` parameter can be `NULL` if not needed.

The `input_from_recorder` function is called with input data to the object. When using live input, it is called with data given to `input`; when playing back, it is called with recorded data. An object should never use its input directly but only what it receives from `input_from_recorder`.

The `playback` function returns true if the recorder is currently playing back recorded data and false if not.

The `dbufbuffer_t` parameters retain their respective ownership. The `input_from_recorder` function must not modify the contents without cloning it first.

```
typedef void (*recorder_input_handler_t)(conf_object_t *obj, dbuffer_t *data,
                                         uint32 uint_data);

SIM_INTERFACE(recorder) {
    recorder_t *(*attach)(conf_object_t *rec, conf_object_t *obj,
                          recorder_input_handler_t input_from_recorder);
    void (*detach)(conf_object_t *rec, recorder_t *r);
    void (*input)(conf_object_t *rec, recorder_t *r,
                  dbuffer_t *data, uint32 uint_data);
    bool (*playback)(conf_object_t *rec);
};

#define RECORDER_INTERFACE "recorder"
```

Execution Context

Execution Context for all methods.

recorder_v2

Implemented By
[recorder](#)

Description

The `recorder_v2` interface is implemented by the recorder, and can be used by any object interacting with the outside world in order to make re-runs of the same simulation behave identically. This is a requirement for reverse execution to work. Objects using this interface must implement the `recorded` interface themselves.

An object uses it by calling the `record` method with itself and the data it wishes to record as parameters. The recorder will then save the data and call the `input` method in the `recorded` interface on the object.

The `playback` method returns whether the recorder is currently playing back recorded data. It may be used by the object to determine if output to the outside world should be dropped or not.

```
SIM_INTERFACE(recorder_v2) {
    void (*record)(conf_object_t *NOTNULL obj,
                   conf_object_t *NOTNULL sender, bytes_t data);
    bool (*playback)(conf_object_t *NOTNULL obj);
};

#define RECORDER_V2_INTERFACE "recorder_v2"
```

The `recorded` interface is implemented by objects that wish to use the `recorder_v2` interface.

The `input` method is called with data that has been recorded. The `playback` parameter is set if the data came from a previous recording, and clear if the it came directly from a call to `record` in `recorder_v2` with live data.

```
SIM_INTERFACE(recorded) {
    void (*input)(conf_object_t *NOTNULL obj, bytes_t data, bool playback);
};

#define RECORDED_INTERFACE "recorded"
```

Execution Context

Execution Context for all methods.

remote_callback

Implemented By
[frontend-server](#)

Description

This interface, implemented by the frontend-server, allows a class to expose a callback that a custom frontend extension may invoke.

The function *register_object_callback* registers a callback. The callback will be known by the name *name* to the front-end. The *target* instance and *user_data* are passed to the callback; the *args* value is defined by the caller. The *has_side_effects* parameter should be `true` if the callback can potentially have any side-effects on the simulation; this is needed to ensure determinism during reverse execution.

The callback function will always be invoked Outside Execution Context.

```
SIM_INTERFACE(remote_callback) {
    void (*register_object_callback)(conf_object_t *frontend_server,
                                    conf_class_t *target,
                                    const char *name,
                                    attr_value_t (*callback)(
                                        conf_object_t *target,
                                        attr_value_t args,
                                        lang_void *user_data),
                                    lang_void *user_data,
                                    bool has_side_effects);
};

#define REMOTE_CALLBACK_INTERFACE "remote_callback"
```

Execution Context

Outside Execution Context for all methods.

rs232_device

Implemented By
[NS16550_c](#)

Description

Currently Simics internal.

```
SIM_INTERFACE(rs232_device) {
    /* Flow control lines */
    void (*cts)(conf_object_t *obj, int status);
    void (*dsr)(conf_object_t *obj, int status);

    /* Ring indicator */
    void (*ring)(conf_object_t *obj, int status);

    /* Carrier detect */
    void (*carrier)(conf_object_t *obj, int status);

    /* Break */
    void (*got_break)(conf_object_t *obj);

    /* Frame error */
    void (*got_frame_error)(conf_object_t *obj);
};

#define RS232_DEVICE_INTERFACE "rs232_device"
```

Execution Context

Instruction Context for all methods.

scale_factor_listener

Implemented By
[frequency_bus](#)

Description

The `scale_factor_listener` interface is used for modeling changes in scale factors. It is mainly used by the `frequency_bus` device, to allow an external device to affect how much the bus scales its input frequency.

The `scale_factor_listener` has exactly the same semantics as the `frequency_listener` interface, the two interfaces only differ in convention: The parameters to `scale_factor_listener.set()` are typically a scale factor rather close to 1, while the parameters of `frequency_listener` represent a base frequency in Hz, which is typically significantly larger than 1000000.

See the documentation on the [frequency_listener](#) interface for more information.

```
SIM_INTERFACE(scale_factor_listener) {
    void (*set)(conf_object_t *NOTNULL obj, uint64 numerator,
                uint64 denominator);
};

#define SCALE_FACTOR_LISTENER_INTERFACE "scale_factor_listener"
```

Execution Context

Instruction Context for all methods.

serial_device

Implemented By

[NS16450](#), [NS16550](#), [NS16550_c](#), [host-serial-console](#), [ser-link-endpoint](#), [telnet_console](#), [text-console](#)

Description

Interface used to connect serial devices together. It can be implemented both by devices such as UARTs and text consoles, and by links.

A character (byte) is sent with the *write()* function; *obj* is the receiving device or link, and *value* is either a data byte, or the out-of-band value `TTY_ABORT`.

The receiver will return the number of characters accepted; i.e. 1 on success and 0 if it could not handle the new byte. If the receiver returns 0, it must later call *receive_ready()* in the sender's `serial_device` interface to signal that new bytes can now be accepted. A serial device must handle the case where the *receive_ready()* function is called although it has no more bytes to send.

```
#define TTY_ABORT      0x100

SIM_INTERFACE(serial_device) {
    int (*write)(conf_object_t *obj, int value);
    void (*receive_ready)(conf_object_t *obj);
};

#define SERIAL_DEVICE_INTERFACE "serial_device"
```

Execution Context

Instruction Context for all methods.

serial_link

Implemented By

[ser-link-endpoint](#), [serial-link](#)

Description

This obsolete interface is implemented by old serial link objects that provide a data link layer interface for streams of characters (bytes). New serial link objects should implement the `serial_device` interface instead.

A device calls the `connect_device()` function to attach itself to the link. The device must implement the `serial_device` interface. The return value is an identification number that should be used in subsequent calls to `send_char()` and `receive_ready()` to identify the device, or -1 if the connection was refused.

The `disconnect_device()` function detaches a serial device from the link. It will not receive any more data from the link and may not call any functions in the interface (except `connect_device()`, if it wishes to connect again).

The `send_char()` function is used by a device to send a character to the link to be delivered to the other device connected to the same link. The character is an integer between 0 and 255. Returns 1 if the character was successfully sent, and 0 if it was dropped.

When a device has returned 0 from its `write()` method in the `serial_device` interface, it must later call the `receive_ready()` function in the link interface to indicate that it is ready to accept new data.

```
SIM_INTERFACE(serial_link) {
    int (*connect_device)(conf_object_t *obj, conf_object_t *NOTNULL dev);
    void (*disconnect_device)(conf_object_t *obj,
                             conf_object_t *NOTNULL dev);
    int (*send_char)(conf_object_t *obj, int id, int ch);
    void (*receive_ready)(conf_object_t *obj, int id);
};

#define SERIAL_LINK_INTERFACE "serial_link"
```

Execution Context

<code>connect_device</code>	Outside execution context
<code>disconnect_device</code>	Outside execution context
<code>send_char</code>	Instruction context
<code>receive_ready</code>	Instruction context

serial_peripheral_interface_slave

Implemented By
[generic_spi_flash](#)

Description

```
SIM_INTERFACE(serial_peripheral_interface_slave) {
    void (*spi_request)(conf_object_t *obj, int first, int last,
                        int bits, dbuffer_t *payload);
    void (*connect_master)(conf_object_t *obj, conf_object_t *master,
                           const char *port,
                           serial_peripheral_interface_flags_t flags);
    void (*disconnect_master)(conf_object_t *obj, conf_object_t *master);
};

#define SERIAL_PERIPHERAL_INTERFACE_SLAVE_INTERFACE \
"serial_peripheral_interface_slave"
```

The `serial_peripheral_interface_slave` interface is implemented by all SPI slave devices.

The `connect_master` and `disconnect_master` functions are used to select which master device is connected. At most one master device can be connected to a slave device at once. The `flags` parameter to `connect_master` should be set according to the documentation of the `serial_peripheral_interface_flags_t` type.

An SPI transfer consists of a number of consecutive calls to the `spi_request` function. The parameters `first` and `last` represent the rise and fall of the Slave Select (SS) pin: `first` is true on the first `spi_request` call of a transfer, while `last` is true on the last call of a transfer.

The `bits` and `payload` parameters describe the data sent by the master device. `bits` defines the number of bits to send, while `payload` defines the data to transfer. The size of the `payload` buffer should be `ceil(bits / 8)` bytes. The first byte corresponds to the first 8 sent bits, and the least significant bit in each byte corresponds to the first sent bit. For example, the 11-bit sequence (first) 1101111101 (last) will be represented as two bytes, 0xfb followed by 0x5.

SEE ALSO

`serial_peripheral_interface_master_interface_t`, `serial_peripheral_interface_flags_t`

Execution Context

Instruction Context for all methods.

signal

Implemented By

[NS16450](#), [NS16550](#), [NS16550_c](#), [PMC1553](#), [etg](#), [eth-transceiver](#), [generic-flash-memory](#), [generic_spi_flash](#), [i2c-bus](#), [ide](#), [mii-management-bus](#), [mii-transceiver](#), [rapidio_simple_device](#), [signal-bus](#), [signal_link_endpoint](#), [signal_to_interrupt](#), [system_panel_bool_out](#)

Description

The signal interface is for modeling a logical signal, such as a reset or interrupt. Signals are always active high in Simics with respect to their function. This may not correspond to the actual electrical level in a real system.

A signal connection has one initiator and one target object, where the initiator calls methods the interface implemented by the target.

The initiator object should have a configuration attribute that pointing to the target object or to an interface port of the target. This attribute, like all other attributes representing object connections, should be of the object kind, or a list of two entries where the first is an object and the second a string representing the name of the port.

The initiator should call *signal_raise()* to raise the signal level to its active level in the target. Once raised, the same initiator may not call *signal_raise()* again without an intervening call to *signal_lower()*. Similarly, an already low signal may not be lowered again by a *signal_lower()* call from the same initiator. The two functions represent the rising and the falling edge of the signal.

The target should handle the case where a signal is lowered directly after it has been raised and treat this call sequence as a valid pulse even within a single clock cycle. The target should also allow the signal to remain raised for some time before it is lowered.

While a target is disconnected, the input signal level is assumed to be low. When an initiator connects to a target by hot plugging, *signal_raise()* should be called if the output signal from the initiator was high. If the signal was low, then no function in the target should be called.

If the signal level is high on disconnect, then the initiator has to call *signal_lower()* before disconnecting from the target. Connect and disconnect is typically done by changing the attribute in the initiator that identifies the target object.

When an initiator is reset, it should call *signal_lower()* if the actual hardware also lowers the output signal on a reset. The target, on the other hand, should not reset its remembered value of the input.

When a connection attribute is restored from a checkpoint or during reverse execution, no function should be called in the signal interface of the target since the actual signal level does not change. The attribute setter code can distinguish between hot-plugging and state restoring by using *SIM_object_is_configured()* and *SIM_is_restoring_state*. See the latter of the two for more documentation.

When an object is first created and the initial signal level is high, the initiator has to call the *signal_raise()* function in the target. This can not be done earlier than in *finalize_instance* (C/C++) or in *post_init()* (DML) since the target has to be fully configured. Again, this should not be done when restoring a checkpoint.

There must only be a single initiator connected to a target, with the exception of the **signal-bus** that may have several initiators.

A target that needs more than one input signal should use ports to implement several signal interfaces.

The *signal_level* method is deprecated and should not be used. It used to be optional and may not even be implemented in old device models.

Note: The `signal` interface should be used instead of the deprecated `simple_interrupt`, `pin` and `reset` interfaces when writing new models.

```
SIM_INTERFACE(signal) {
    void (*signal_raise)(conf_object_t *NOTNULL obj);
    void (*signal_lower)(conf_object_t *NOTNULL obj);
    /* signal_level is deprecated */
#ifndef SIMICS_4_0_API || defined SHOW_OBSOLETE_API
    void (*signal_level)(conf_object_t *NOTNULL obj, int level);
#else
    void (*deprecated_signal_level)(conf_object_t *NOTNULL obj, int level);
#endif
};

#define SIGNAL_INTERFACE "signal"
```

Execution Context

Instruction Context for all methods.

simple_dispatcher

Implemented By

[cell](#), [frequency_bus](#)

Description

The `simple_dispatcher` interface is used for subscribing to changes of some monitored state. The interface is currently used for subscribing to changes in frequencies and to changes in frequency scale factors, via the interfaces `frequency_listener` and `scale_factor_listener`, respectively.

A device subscribes with the `subscribe` function. Before this function returns, it should send a signal to the calling object, telling the current state. After this, signals should be sent whenever the monitored state changes.

The function `unsubscribe` should be used to unsubscribe from monitored changes. A listener may not subscribe or unsubscribe twice on the same port.

```
SIM_INTERFACE(simple_dispatcher) {
    void (*subscribe)(conf_object_t *NOTNULL bus,
                     conf_object_t *NOTNULL listener,
                     const char *listener_port);
    void (*unsubscribe)(conf_object_t *NOTNULL bus,
                        conf_object_t *NOTNULL listener,
                        const char *listener_port);
};

#define SIMPLE_DISPATCHER_INTERFACE "simple_dispatcher"
```

Execution Context

Outside Execution Context for all methods.

simple_interrupt

Implemented By
[CL-PD6729, SIL680A](#)

Description

A device calls *interrupt()* to logically raise an interrupt and *interrupt_clear()* to lower an interrupt.

The first argument is the object to interrupt, usually a cpu. The integer argument to both functions may indicate an interrupt level or interrupt pin depending on the receiving device.

On ARM the integer argument indicates whether the interrupt is normal or fast, by being either ARM_INT_IRQ or ARM_INT_FIQ defined by the ARM API (by including <simics/arch/arm.h>).

Note: Obsoleted by the signal interface.

```
SIM_INTERFACE(simple_interrupt) {
    void (*interrupt)(conf_object_t *NOTNULL, int);
    void (*interrupt_clear)(conf_object_t *NOTNULL, int);
};

#define SIMPLE_INTERRUPT_INTERFACE "simple_interrupt"
```

Execution Context

Instruction Context for all methods.

slaver_message

Implemented By
[slaver](#)

Description

This interface is implemented by the slaver, and is used from a slave agent to send data to the slave.

The *send* method sends a deterministic message to the slave to be delivered at *time*, ordered by *skey*.

The *send_async* method sends a non-deterministic (asynchronous) message to the slave to be delivered as soon as possible.

```
SIM_INTERFACE(slaver_message) {
    void (*send)(conf_object_t *obj,
                 slave_time_t time, uint64 skey, bytes_t msg);
    void (*send_async)(conf_object_t *obj, bytes_t msg);
};

#define SLAVER_MESSAGE_INTERFACE "slaver_message"
```

Execution Context

Execution Context for all methods.

snoop_memory

Implemented By

[mem-traffic-meter](#), [state-assertion](#), [trace-mem-hier](#)

Description

This interface is described with the `timing_model` interface.

Execution Context

Instruction Context for all methods.

software

Implemented By
[software_tracker](#)

Description

```
typedef enum {
    Sim_Prop_Modified,
    Sim_Prop_Added,
    Sim_Prop_Removed
} property_status_change_t;

SIM_INTERFACE(software) {
    uint64 (*root_node_id)(conf_object_t *NOTNULL obj);
    attr_value_t (*get_node)(conf_object_t *NOTNULL obj, uint64 node_id);
    attr_value_t (*get_subtree)(conf_object_t *NOTNULL obj, uint64 node_id);
    attr_value_t (*get_current_nodes) (
        conf_object_t *NOTNULL obj, conf_object_t *cpu);
    attr_value_t (*notify_create) (
        conf_object_t *NOTNULL obj, uint64 node_id, bool recursive,
        void (*callback)(cbdata_call_t data, conf_object_t *obj,
                         conf_object_t *cpu, uint64 node_id),
        cbdata_register_t data);
    attr_value_t (*notify_destroy) (
        conf_object_t *NOTNULL obj, uint64 node_id, bool recursive,
        void (*callback)(cbdata_call_t data, conf_object_t *obj,
                         conf_object_t *cpu, uint64 node_id),
        cbdata_register_t data);
    attr_value_t (*notify_property_change) (
        conf_object_t *NOTNULL obj, uint64 node_id, const char *key,
        bool recursive,
        void (*callback) (
            cbdata_call_t data, conf_object_t *obj,
            conf_object_t *cpu, uint64 node_id, const char *key,
            attr_value_t old_val, attr_value_t new_val,
            property_status_change_t property_status),
        cbdata_register_t data);
    attr_value_t (*notify_cpu_move_from) (
        conf_object_t *NOTNULL obj, uint64 node_id,
        void (*callback)(cbdata_call_t data, conf_object_t *obj,
                        conf_object_t *cpu, attr_value_t node_path),
        cbdata_register_t data);
    attr_value_t (*notify_cpu_move_to) (
        conf_object_t *NOTNULL obj, uint64 node_id,
```

```

        void (*callback)(cbdata_call_t data, conf_object_t *obj,
                         conf_object_t *cpu, attr_value_t node_path),
        cbdata_register_t data);
attr_value_t (*notify_syscall) (
    conf_object_t *NOTNULL obj, uint64 node_id, bool recursive,
    void (*callback) (
        cbdata_call_t data, conf_object_t *obj,
        conf_object_t *cpu, uint64 node_id,
        const char *syscall),
        cbdata_register_t data);
attr_value_t (*notify_callbacks_done) (
    conf_object_t *NOTNULL obj, uint64 node_id,
    void (*callback)(cbdata_call_t data, conf_object_t *obj),
    cbdata_register_t data);
attr_value_t (*notify_after_callbacks_done) (
    conf_object_t *NOTNULL obj, uint64 node_id,
    void (*callback)(cbdata_call_t data, conf_object_t *obj),
    cbdata_register_t data);
void (*cancel_notify) (
    conf_object_t *NOTNULL obj, uint64 callback_id);
bool (*set_property) (
    conf_object_t *NOTNULL obj, uint64 node_id,
    const char *key, attr_value_t value);
attr_value_t (*request)(conf_object_t *NOTNULL obj);
attr_value_t (*release)(conf_object_t *NOTNULL obj);
};

#define SOFTWARE_INTERFACE "software"

```

root_node_id returns the ID of the root node. This ID never changes.

get_node returns a dictionary for a given node, containing all the node's properties; or nil if no such node exists.

get_subtree returns a dictionary mapping node IDs to node dictionaries; it contains the specified node and all its descendants. If the specified node does not exist, the return value is nil.

get_current_nodes returns the current node path of the given processor. (That is, a list of node IDs of all nodes that are currently running on the processor, the first node being the root node, and every subsequent node a child of the node before it.) If the tracker is not tracking the given processor, the return value is nil.

notify_create and *notify_destroy* register callbacks to be called when the node is created and destroyed, respectively. It is safe to read the node with *get_node* from within the callback function.

notify_property_change registers a callback that is triggered when the given property changes on the node (or any property, if *key* is NULL).

If recursive, the callback will be triggered for the given node's descendants as well as the node itself. All callbacks receive a *cpu* argument; it specifies the processor that caused the event, but may be `NULL` if the event was not caused by a processor.

notify_cpu_move_from and *notify_cpu_move_to* register callbacks that are triggered when a processor moves from one node path to another—but only if either path lies in the subtree rooted at the given node.

notify_syscall registers a callback that will run whenever the software, represented by a node, makes a system call. The *obj* argument is the software tracker object. When registering the callback, the *node_id* argument is the identifier for the node who's system calls should result in a call to the registered callback.

The *recursive* argument is used to tell the framework if system calls made by any of the children (or their children and so on) to the specified node should generate a callback or not. Trackers will associate the system call with the lowest entity of software, often referred to as a thread. If system calls generated by only one single thread is interesting then the *recursive* argument is not needed. However, if system calls for a process or the entire system is interesting, then the *recursive* argument must be specified. This is required since the process node itself, nor the system node will be associated with any system call, only the thread will. The *(*callback)* argument is the callback function that should be called upon a system call. The *data* argument will be passed back to the callback each time, and is not used by the framework. It can be left as `NULL` if the callback does not need it. The callback function must take five parameters. The first argument is the *data* that is the same as the *data* used when registering the callback function. The *obj* argument is the software tracker object. The processor that the tracker identified as the source for the system call will be given by the *cpu* argument. The identification of the node that represent the software that issued the system call will be given by *node_id* argument. Finally the *syscall* argument represents the name of the system call, or if the tracker failed to convert the system call into a name it will contain the system call number prefixed with `sys_`, such as `sys_123`.

Since a single update to the node tree can result in several different callbacks being triggered, reading nodes with *get_node* from a callback function may yield a result containing updates whose callbacks have not yet been run. (For example, if two nodes change their `name` attributes simultaneously, the final state of both nodes may be visible to both property change callbacks.) With *notify_callbacks_done*, you can register a callback that will run when all other callbacks pertaining to a particular change in the node tree are finished. Note that it is not possible to modify any node property at this stage.

With *notify_after_callbacks_done*, you can register a callback that will run when all other callbacks pertaining to a particular change in the node tree and all *notify_callbacks_done* are finished. At this stage it is possible to modify node properties. Note though that other callbacks might have already run at this stage, which might have already changed node properties.

The functions that install callbacks return an integer ID, or nil on error. This ID can be passed to *cancel_notify* in order to uninstall the callback.

set_property sets a property on a node, and returns true if successful. Note that not all node properties are writeable.

request is used to register users that are interested in using the tracker framework and activates the tracker framework if it is not yet activated. It returns a text string upon errors that describes the error, otherwise nil.

release will decrease the number of users for each call and when no more users are interested in the tracker framework the trackers framework will be disabled. It returns a text string upon errors that describes the error, otherwise nil.

Execution Context

Instruction Context for all methods.

symtable

Implemented By
[symtable](#)

Description

Note: This is a non-canonical interface, where some function members do not take a pointer to the exporting object as the first argument.

This is a collection of API calls for the symtable module, providing symbolic debugging support of the target machine.

```
SIM_INTERFACE(symtable) {
    attr_value_t (*eval_sym)(conf_object_t *cpu,
                           const char *expr,
                           attr_value_t *frame,
                           const char *format);
```

Evaluates the symbolic expression *expr* in the given stack frame on the specified cpu. The result is the list (*value*, *type*) where *value* depends on the format, and *type* is a string describing the type of the result. If *format* is “v”, then *value* is the value of the result as a string or integer; otherwise it is a human-readable string representation of the result. The only operators allowed in *expr* are casts, indirection and member selection (no arithmetic).

```
attr_value_t (*stack_trace)(conf_object_t *cpu,
                           int maxframes);
```

Does a stack trace in the current context of the specified cpu, and returns a list of stack frames (at most *maxframes*). Each stack frame is in turn a list of state variables, suitable for passing to eval_sym above, but has no defined structure other than that the first element is the program counter. This program counter points to the next instruction to be executed in respective frame.

```
attr_value_t (*fun_args)(conf_object_t *cpu,
                        attr_value_t *frame);
```

Given a stack frame, returns a list of the formal and actual parameters to the function where that frame is currently executing. Each element of the returned list is a list on the form (*name*, *value*).

```
attr_value_t (*source_profile)(conf_object_t *symtable,
```

```
conf_object_t *profile,  
int view);
```

Find the corresponding source lines for profile values and return a list on the form
((*source-file*, (*line*, *value*) ...) ...).

```
};
```

Execution Context

Instruction Context for all methods.

timing_model

Implemented By

[g-cache](#), [id-splitter](#), [state-assertion](#), [trace-mem-hier](#), [trans-sorger](#), [trans-splitter](#), [trans-staller](#)

Description

The `timing_model` interface is used to communicate stall times for memory accesses. It is typically exported by cache models. The `operate()` function is then called on every memory access that misses in the STC, and the return value from the call is the number of cycles to stall.

The `snoop_memory` interface has the exact same layout as the `timing_model` interface, but its `operate()` function is called after the memory access has been performed. The return value from the `operate()` function of a `snoop_memory` interface is ignored.

See the *Extension Builder User Guide* for more information on how to use these interfaces.

```
SIM_INTERFACE(timing_model) {
    cycles_t (*operate)(conf_object_t *NOTNULL mem_hier,
                        conf_object_t *NOTNULL space,
                        map_list_t *NOTNULL map_list,
                        generic_transaction_t *NOTNULL mem_op);
};

SIM_INTERFACE(snoop_memory) {
    cycles_t (*operate)(conf_object_t *NOTNULL mem_hier,
                        conf_object_t *NOTNULL space,
                        map_list_t *NOTNULL map_list,
                        generic_transaction_t *NOTNULL mem_op);
};

#define TIMING_MODEL_INTERFACE "timing_model"
#define SNOOP_MEMORY_INTERFACE "snoop_memory"
```

Execution Context

Instruction Context for all methods.

translate

Implemented By

[bitmask-translator](#), [flash-memory](#), [generic-flash-memory](#), [memory-space](#)

Description

The `translate` interface is implemented by objects that bridge between memory spaces. `translate()` may change the physical address of `mem_op`, but not its size. The return value from `translate()` can be a target memory space or NULL in which case the space provided in the configuration is used.

Note: The initiator of a memory operation may assume that the result of the `translate()` method does not change between invocations. This enables some performance optimizations which are the main motivation of the interface. In order to handle changing translations correctly and efficiently, knowledge of Simics internals is typically needed. A call to `SIM_mem_op_ensure_future_visibility()` on the memop ensures correctness, but it also eliminates the performance benefits of translators.

```
SIM_INTERFACE(translate) {
    conf_object_t *(*translate)(conf_object_t *NOTNULL obj,
                               generic_transaction_t *NOTNULL mem_op,
                               map_info_t mapinfo);
};

#define TRANSLATE_INTERFACE "translate"
```

Execution Context

Instruction Context for all methods.

uint64_state

Implemented By

[etg](#), [system_panel_number_out](#)

Description

Interface to transfer a state representable in an uint64 from one device to another.
Examples of what the state might represent includes:

- a fixed-point value representing the level of an analog signal
- an integer representing a counter
- an integer representing an enum value

The initiator should call *set* when the value changes, and after a new target is connected.
Multiple calls with the same level should be accepted by the signal receiver.

A device implementing this interface may choose to only accept a certain set of integer values; it is then an error to send any other values to the *set()* method. A user must therefore be careful to read the documentation of both the source and destination object to make sure they are compatible.

No interface call needs to be done after disconnecting a target; the target needs to be notified of this through some other channel (typically via a connector)

Note: The `uint64_state` interface should be used instead of the deprecated `multi_level_signal` interface when writing new models.

```
SIM_INTERFACE(uint64_state) {
    void (*set)(conf_object_t *NOTNULL obj, uint64 level);
};

#define UINT64_STATE_INTERFACE "uint64_state"
```

Execution Context

Instruction Context for all methods.

x86_cpuid

Implemented By
[clipboard-gateway](#)

Description

```
SIM_INTERFACE(x86_cpuid) {
    cpuid_ret_t (*cpuid)(conf_object_t *obj, conf_object_t *cpu,
                         uint32 in_eax, uint32 in_ebx, uint32 in_ecx,
                         uint32 in_edx);
};

#define X86_CPUID_INTERFACE "x86_cpuid"
```

The CPUID interface makes it possible to customize responses to CPUID requests. The *cpuid* method should set taken to nonzero if it managed to handle the request, zero otherwise. When taken is non-zero, then the returned values in out_a, out_b, out_c, and out_d will be written to the first four general purpose registers.

Execution Context

Instruction Context for all methods.

Chapter 7

Haps

Central_Blocked

Provided By
[central](#)

Callback Type

```
void (*)(lang_void *callback_data,  
        conf_object_t *trigger_obj);
```

Description

Execution is blocked because the central client is waiting for other simulation nodes to catch up.

Central_Unblocked

Provided By
[central](#)

Callback Type

```
void (*)(lang_void *callback_data,  
        conf_object_t *trigger_obj);
```

Description

A blocking condition, as signalled by the Central_Blocked hap, has been removed, and simulation may continue.

CLI_Command_Added

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj, char *command_name);
```

Description

Triggered when a CLI command is defined.

CLI_Variable_Write

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj, char *variable_name);
```

Description

Deprecated

Component_Change

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj);
```

Description

Internal: Similar to Component_Hierarchy_Change but also triggered for components that are not part of any complete hierarchy including non-instantiated components.

Component_Hierarchy_Change

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj,  
         conf_object_t *top_level_component);
```

Description

Internal: Triggered when an instantiated component hierarchy is modified. The hap is associated with the top-level component of the modified hierarchy.

Core_Address_Not_Mapped

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj,
          integer_t physical_address, integer_t access_type,
          integer_t size);
```

Description

Triggered when an access to a memory-space has no target. The default handler for this trap will raise an exception, returning the simulation to the prompt unless the attribute sim->handle_outside_memory is set to TRUE.

Core_Asynchronous_Trap

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, integer_t trap_number);
```

Index

trap_number

Description

SPARC: Triggered when an asynchronous trap occurs. This is either an external interrupt *trap number* == 0x60 or interrupt level n *trap number* 0x41 - 0x4F, or an asynchronous trap initiated by the user calling trap_cpu in the sparc-interrupt interface. The Core_External Interrupt can also be used to catch interrupts, the difference is that Core_Asynchronous_Trap is only triggered if interrupts are enabled.

x86: Triggered when an asynchronous trap occurs. This is an *interrupt vector*. Core_Asynchronous_Trap is only triggered if interrupts are enabled.

Core_At_Exit

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj);
```

Description

Triggered at exit. No parts of the Simics API may be called by the callbacks.

Core_Back_To_Front

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj);
```

Description

Deprecated.

Core_Breakpoint_Change

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj);
```

Description

Triggered on an object when breakpoints attached to that object are inserted, deleted or changed in any way.

Core_Breakpoint_Memop

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj,
          integer_t breakpoint_number,
          generic_transaction_t *memop);
```

Index

breakpoint_number

Description

Triggered when a breakpoint is triggered. *breakpoint_number* is the breakpoint number (as returned by *SIM_breakpoint*). If there are multiple breakpoints on an instruction then all installed haps will be run before control is transferred to the frontend (when applicable). The supplied *memop* can be used to figure out details about the transaction.

Core_Conf_Class_Register

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, char *class_name);
```

Description

Triggered when a new configuration class has been registered. Called in outside execution context.

Core_Conf_Class_Unregister

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, char *class_name);
```

Description

Triggered when a configuration class has been unregistered. Called in outside execution context.

Core_Conf_Clock_Change_Cell

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj,
          conf_object_t *old_cell, conf_object_t *new_cell);
```

Description

Triggered when a clock object changes cell

Core_Conf_Object_Change_Clock

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj,
```

```
conf_object_t *old_clock,  
conf_object_t *new_clock);
```

Description

Triggered when an object's reference clock is changed

Core_Conf_Object_Create

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj);
```

Description

Triggered when a new configuration object's *init_object* method returns successfully, but before the object's attributes are set and before the object's *finalize_instance* method is called. Since the object is not fully created hap callbacks must not access any of the object's attributes or call any methods on the object's interfaces. Callbacks are called in outside execution context.

Core_Conf_Object_Created

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj);
```

Description

Triggered when a configuration object has been created and finalized. This hap will always be followed by a Core_Conf_Objects_Created hap, but this hap can be triggered for more than one object before the Core_Conf_Objects_Created hap.

Core_Conf_Object_Delete

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj, char *object_name);
```

Description

Triggered after a configuration object has been deleted. Called in outside execution context.

Core_Conf_Object_Pre_Delete

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,  
        conf_object_t *trigger_obj);
```

Description

Triggered just before a configuration object is deleted. Called in outside execution context.

Core_Conf_Object_Rename

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,  
        conf_object_t *trigger_obj, char *old_name);
```

Description

Triggered after an object's name has changed. Called in outside execution context.

Core_Conf_Objects_Created

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,  
        conf_object_t *trigger_obj);
```

Description

Triggered if and only if at least one Core_Conf_Object_Created hap has been triggered.

Core_Conf_Objects_Deleted

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj);
```

Description

Triggered when one or more objects have been deleted from the configuration after the Core_Conf_Object_Delete hap has been triggered for all of the objects. Called in outside execution context.

Core_Configuration_Loaded

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj);
```

Description

Triggered when a configuration has been loaded. This hap can be triggered several times during a session since it is possible to append a new configuration to the existing one. In most cases it is better to use the *finalize_instance* function in the class_data_t instead. That function is called when the object creation is finished. Callbacks called in outside execution context.

Core_Context_Activate

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj,  
         conf_object_t *other_ctx, conf_object_t *cpu);
```

Description

Triggered when this context replaces another context as the current context of a processor.

Core_Context_Change

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj, conf_object_t *cpu);
```

Description

Triggered when the context is set to the current context for a processor.

Core_Context_Deactivate

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,
         conf_object_t *trigger_obj,
         conf_object_t *other_ctx, conf_object_t *cpu);
```

Description

Triggered when another context replaces this context as the current context of a processor.

Core_Context_Updated

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,
         conf_object_t *trigger_obj);
```

Description

Triggered when the context is updated in some way, for example when a new symtable is associated with the context.

Core_Continuation

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,
         conf_object_t *trigger_obj);
```

Description

Triggered at the (re)start of the simulation. The `Core_Simulation_Stopped` hap is called when the simulation is stopped. Callbacks called in outside execution context.

Core_Control_Register_Read

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj,
          integer_t register_number);
```

Index

register_number

Description

Triggered when a control register is read. The hap is called before the read is performed; thus no registers have been modified.

Note that if the callback routine breaks to the frontend by raising an exception, the instruction will be replayed possibly causing repeated memory operations.

For x86 processors, this hap is triggered when a control register is read using a `mov` or `smsw` instruction (i.e. only explicit reads).

For PowerPC processors, this hap is triggered by the `mf spr`, `mf msr` `mf sr` and `mf srin` instructions (i.e. only explicit reads).

Core_Control_Register_Write

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj,
          integer_t register_number, integer_t value);
```

Index

register_number

Description

Triggered when a control register is written. Note that *value* is not (necessarily) the final contents in the register. When the callback is called, the processor register has not yet been updated with the new value.

If the callback routine breaks to the frontend by raising an exception, the instruction will be replayed possibly causing repeated memory operations.

For x86 processors, this hap is triggered by `clts`, `lmsw`, and `mov`. Page fault updates of CR2 should be caught with the `Core_Exception` hap.

For PowerPC processors, this hap is triggered by the `mt spr`, `mt msr`, `mt sr` and `mt srin` instructions (i.e. only explicit writes).

Core_Cycle_Count

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj,
          integer_t current_cycle_count);
```

Index

current_cycle_count

Description

Deprecated - Use event system instead.

Core_Device_Access_Memop

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj,
          generic_transaction_t *memop);
```

Description

Triggered when a device access is performed.

Core_Disable_Breakpoints

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, int reenable);
```

Description

Disable (or reenable) all breakpoints. This hap is primarily used to improve reverse performance in certain situations.

Core_Discard_Future

Provided By
[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,  
        conf_object_t *trigger_obj);
```

Description

Triggered when recorded events should be forgotten. Called in outside execution context.

Core_DSTC_Flush_Counter

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,  
        conf_object_t *trigger_obj, integer_t type,  
        integer_t virtual_address,  
        integer_t physical_address, integer_t counter);
```

Description

Triggered when the DSTC flushes a line's hit counter. It reports how many hits the STC recorded since the line was inserted.

Core_Exception

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,  
        conf_object_t *trigger_obj,  
        integer_t exception_number);
```

Index

exception_number

Description

Triggered when an exception/trap is taken by a processor. This happens after exception specific side effects have taken place, such as register updates, but before any control transfer.

The exact meaning of the exception number depends on the simulated processor architecture, the function *exception->get_name* gives its symbolic name. *exception->get_number* translates from symbolic name to exception number.

Core_Exception_Return

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj,
          integer_t exception_number);
```

Index

exception_number

Description

Triggered when an exception/trap handler finishes execution. The hap is triggered before any processor state has changed.

The following instructions triggers this hap (by processor class):

MIPS: eret and deret

PowerPC (32): rfi and rfci

PowerPC (64): rfi, rfid and hrfid

SH: rte

SPARC: done and retry

TMS320C64: b.s2 irp

x86/x86-64: The iret family.

The *exception_number* parameter is only valid for SPARC processors.

Core_External_Interrupt

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, integer_t source_mid);
```

Index

source_mid

Description

(SPARC only) Triggered when a processor receives an external interrupt. The trigger object is the receiving cpu, and *source_mid* is the mid of the sending cpu/device. The hap will be triggered even if interrupts are disabled, but not if the interrupt logic is busy.

Core_Frequency_Changed

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, integer_t old_freq,
          integer_t new_freq);
```

Description

Triggered when the frequency of a cycle queue has changed. Parameters are the old and new frequencies in Hz.

Core_Global_Message

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, char *message);
```

Description

Experimental

Core_Hap_Callback_Installed

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, integer_t hap_number,
          integer_t range_low, integer_t range_high);
```

Index

hap_number

Description

Triggered when a callback is installed on a hap. The callback called by this hap is not allowed to install any hap callbacks.

Core_Hap_Callback_Removed

Provided By
[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,
         conf_object_t *trigger_obj, integer_t hap_number,
         integer_t range_low, integer_t range_high);
```

Index

hap_number

Description

Triggered when a callback is unregistered from a hap. The callback called by this hap is not allowed to remove any hap callbacks.

Core_Hap_Type_Added

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,
         conf_object_t *trigger_obj, char *hap_name);
```

Description

Triggered when a new hap type is added to the simulator. The hap is not triggered for the initial set of core haps. Callbacks called in outside execution context.

Core_Image_Activity

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,
         conf_object_t *trigger_obj, int type, int onoff);
```

Description

Triggered on I/O activity in an image object.

Core_Initial_Configuration

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,
         conf_object_t *trigger_obj);
```

Description

Deprecated - Use the Core_Configuration_Loaded hap instead.

Core_Log_Message

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, int type,
          char *message);
```

Index

log type

Description

Triggered when a log message is registered that is supposed to be logged. I.e. with matching type and group(s) and level.

Core_Log_Message_Extended

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, int type,
          char *message, int level, int group);
```

Description

Triggered when a log message is registered that is supposed to be logged according to the log level.

Core_Magic_Instruction

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, integer_t parameter);
```

Index

parameter

Description

Triggered when Simics executes a magic instruction. The parameter is taken from the instruction and is architecture-dependent.

Core_Memory_Space_Map_Changed

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,
         conf_object_t *trigger_obj);
```

Description

Experimental

Core_Mode_Change

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,
         conf_object_t *trigger_obj, integer_t old_mode,
         integer_t new_mode);
```

Description

Triggered when a processor changes privilege mode for whatever reason; usually an exception or return from an exception. The hap occurs after the processor has changed mode but before any instructions have been executed in the new mode.

For x86 processors, the modes are Sim_CPU_Mode_User for CPL 3, and Sim_CPU_Mode_Supervisor for CPL 0-2.

For other processors, the modes are Sim_CPU_Mode_User or Sim_CPU_Mode_Supervisor. The UltraSPARC T1 processor also has the Sim_CPU_Mode_Hypervisor mode.

Core_Module_Loaded

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,
         conf_object_t *trigger_obj, char *module_name);
```

Description

Triggered when a module is loaded into Simics. Called in outside execution context.

Core_Multithreading_Changed

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, int on/off);
```

Description

Triggered when multithreaded simulation is enabled or disabled.

Core_NotImplemented

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, int line, char *file,
          char *rcsid, char *message, integer_t data);
```

Description

Triggered when the simulator encounters unimplemented functionality.

Core_Periodic_Event

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj,
          integer_t cycles_between_events);
```

Index

cycles_between_events

Description

Deprecated - Use event system instead.

Core_Preferences_Changed

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj);
```

Description

Triggered when an attribute in the prefs object is written. Called in outside execution context.

Core_Processor_Schedule_Changed

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj);
```

Description

Triggered when the scheduling order of the processors has changed. Called in outside execution context.

Core_Recent_Files_Changed

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj);
```

Description

Triggered when the list of recently used files has changed. Called in outside execution context.

Core_Rexec_Active

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj, int active_flag);
```

Description

Triggered when reverse execution support is activated or deactivated. Called in outside execution context.

Core_Simulation_Mode_Change

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, int simulation_mode);
```

Index

simulation_mode

Description

Triggered when the simulation mode for the processor has changed.

Core_Simulation_Stopped

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, integer_t exception,
          char *error_string);
```

Index

exception

Description

Triggered when the simulation stops. *exception* will always be SimExc_No_Exception, and *error_string* will always be NULL. After this hap, simulation will not advance (triggering Core_Continuation) until *SIM_continue()* is called again. Callbacks called in outside execution context.

Core_SkipTo_Progress

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, int progress);
```

Description

Triggered with an estimate of skip-to progress. Called in outside execution context.

Core_Source_Step

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,
         conf_object_t *trigger_obj, conf_object_t *cpu);
```

Description

Triggered when source code stepping is done.

Core_Step_Count

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,
         conf_object_t *trigger_obj,
         integer_t current_step_count);
```

Index

current_step_count

Description

Deprecated - Use event system instead.

Core_Sync_Instruction

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,
         conf_object_t *trigger_obj, integer_t type);
```

Index

type

Description

Triggered when an synchronizing instruction is executed. The *type* parameter describe what kind of instruction is executing. Its encoding is specific to each architecture.

For SH, this hap is triggered when a synco instruction is executed. The type contains 0.

For SPARC-V9, this hap is triggered when a membar or stbar instruction is executed. For membar, the type contains the 7-bit field cmask|mmask specified in the instruction. For stbar, the type is 8 (equivalent to membar #StoreStore).

For x86, this hap is triggered when a fence instruction is executed. The type is set from the list provided by the x86_instruction_sync_type_t enum.

Core_Time_Transition

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, int in_the_past);
```

Description

Triggered whenever there is a transition between the past and the present. Called in outside execution context.

Core_Timing_Model_Change

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj);
```

Description

Triggered on an object when a timing model or snoop device is inserted, deleted, or changed.

Core_Workspace_Changed

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj);
```

Description

Triggered when a new Simics workspace directory is selected. Called in outside execution context.

Core_Write_Configuration

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, char *file_name);
```

Description

Deprecated.

Ethernet_Frame

Provided By
[ethernet-link](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, integer_t timestamp);
```

Description

This hap is only triggered by the old ethernet-link. The new Ethernet links use an ethernet probe instead.

Triggered when a frame is sent over an ethernet network. The network object is given as a parameter. The contents of the packet can be retrieved or modified using the `last_frame` attribute. Setting `last_frame` to nil will drop the packet from the network, and modifying it will deliver a modified packet to the destination. The `timestamp` argument is the current time in nanoseconds.

FC_SCSI_Command

Provided By
[fc-disk](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj,
          integer_t command_number, integer_t start,
          integer_t len);
```

Index

`command_number`

Description

Triggered when a SCSI command is received by a SCSI device. The parameters `start` and `len` are only used for read and write commands.

Firewire_Reset

Provided By

[firewire-bus](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj);
```

Description

Triggered when the bus is reset. It is invoked after calculating the default topology. During the hap the self_ids attribute can be used to change the self id packets sent to the devices on the bus. The connected_devices attribute can also be changed to modify the mapping from physical id to device.

Firewire_Transfer

Provided By

[firewire-bus](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj);
```

Description

Triggered when an packet travels through a firewire bus. During the hap handler the current_transfer attribute of the bus can be used to inspect and/or change the current transfer. If you set it to NULL the transfer is aborted and Firewire_V2_Ack_No_Ack is returned to the initiator of the transfer.

Generic_Message_Frame

Provided By

[g-link](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj, integer_t timestamp);
```

Description

Triggered when a frame is sent over a generic message network. The network object is given as a parameter. The contents of the packet can be retrieved/modified using the first element in the pending_frames attribute.

Gfx_Break_String

Provided By

[x11-console](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, integer_t break_id);
```

Index

break_id

Description

Triggered when a graphics breakpoint matches the screen. *break_id* is the number returned when a breakpoint is set.

Graphics_Console_New_Title

Provided By

[x11-console](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, char *title);
```

Description

Triggered when the console is assigned a new title.

Graphics_Console_Show_Hide

Provided By

[x11-console](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, int state);
```

Description

Triggered when the console is shown or hidden. The *hap* argument is 1 when the console is shown and 0 when hidden.

Internal_Bookmark_List_Changed

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,  
        conf_object_t *trigger_obj);
```

Description

Simics internal. Triggered when a bookmark has been added or deleted.

Internal_Micro_Checkpoint_Loaded

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,  
        conf_object_t *trigger_obj);
```

Description

Simics internal

Internal_SB_Barrier

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,  
        conf_object_t *trigger_obj);
```

Index

barrier

Description

Simics internal

Internal_SB_Command_File

Provided By

[Simics Core](#)

Callback Type

```
void (*)(lang_void *callback_data,  
        conf_object_t *trigger_obj);
```

Index

id

Description

Simics internal

Internal_SB_Cycle_Count

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj);
```

Index

id

Description

Simics internal

Internal_SB_Log

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj);
```

Index

id

Description

Simics internal

Internal_SB_Pipe

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj);
```

Index

pipe

Description

Simics internal

Internal_SB_Step_Count

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj);
```

Index

id

Description

Simics internal

Internal_SB_Time

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj);
```

Index

id

Description

Simics internal

Internal_SB_Variable

Provided By
[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,  
         conf_object_t *trigger_obj);
```

Index

pipe

Description

Simics internal

Internal_Time_Direction_Changed

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, int rev);
```

Description

Simics internal. 'rev' is 1 when reverse execution is initiated, and 0 when reverse execution stops. In addition, a number of haps with rev = -1 may be fired during reverse execution at reverse chronologically ordered moments. This is typically used to update the user interface during reverse execution.

Internal_Time_Quantum_Changed

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj);
```

Description

Simics internal. Triggered when the time quantum has changed.

Rexec_Limit_Exceeded

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, int limit_type);
```

Index

limit_type

Description

Triggered when a reverse execution resource limit is exceeded.

SCSI_CDROM_Command

Provided By

[scsi-cdrom](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj,
          integer_t command_number, integer_t start,
          integer_t len);
```

Index

command_number

Description

Triggered when a SCSI command is received by a SCSI device. The parameters *start* and *len* are only used for read and write commands.

SCSI_Disk_Command

Provided By

[simple-scsi-disk](#), [scsi-disk](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj,
          integer_t command_number, integer_t start,
          integer_t len);
```

Index

command_number

Description

Triggered when a SCSI command is received by a SCSI device. The parameters *start* and *len* are only used for read and write commands.

Symtable_Updated

Provided By

[symtable](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj);
```

Description

Triggered when new symbols have been added to a symtable.

Text_Console_New_Title

Provided By

[xterm-console](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, char *new_title);
```

Description

Triggered when the console is assigned a new title.

Text_Console_Show_Hide

Provided By

[xterm-console](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, int is_shown);
```

Description

Triggered when the console is shown or hidden. *is_shown* is 1 when the console is shown and 0 when hidden.

UI_Run_State_Changed

Provided By

[Simics Core](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, char *state);
```

Description

Triggered when the run state changes. The argument is one of: Stopped - simulation stopped and may not run, Stopped_Fwd - stopped and may run forward, Stopped_Fwd_Rev - stopped and may run forward or reverse, Forwarding - simulation is running forward, Forwarding_Rev - running forward and may run in reverse, Reversing - simulation is running in the reverse direction

Vga_Break_String

Provided By

[accel-vga, vga-pci](#)

Callback Type

```
void (*) (lang_void *callback_data,
          conf_object_t *trigger_obj, char *string);
```

Index

string_id

Description

Triggered when output matches a string set to break on. The *string_id* is the id returned by the vga_add_string_notification method in the vga_text interface.

Xterm_Break_String

Provided By

[xterm-console](#), [telnet-console](#)

Callback Type

```
void (*)(lang_void *callback_data,  
        conf_object_t *trigger_obj, char *break_string);
```

Index

break_id

Description

Triggered when the output matches a string set to break on. The *break_id* is the number returned when a string breakpoint is set.

Chapter 8

Modules

accel-vga

Filename

accel-vga.so

Classes

[accel-vga](#)

Haps

[Vga_Break_String](#)

agent_symdebug_interface

Filename

agent_symdebug_interface.so

AM79C973

Filename

AM79C973.so

Classes

[AM79C973](#)

AT24Cxx

Filename

AT24Cxx.so

Classes[AT24Cxx](#)**attr-meter****Filename**

attr_meter.pymod

Classes[attr-meter](#)**Global Commands**[new-attr-meter](#) create a new attribute meter**BCM5703C****Filename**

BCM5703C.so

Classes[BCM5703C](#)**BCM5704C****Filename**

BCM5704C.so

Classes[BCM5704C](#)**bitmask-translator****Filename**

bitmask-translator.so

Classes[bitmask-translator](#)**breakpoint-manager**

Filename

breakpoint-manager.so

central**Filename**

central.so

Classes

[central-client](#)
[central-server](#)

Haps

[Central_Blocked](#)
[Central_Unblocked](#)

Global Commands

[connect-central](#) connect to Simics Central
[new-central-server](#) create a Simics Central server

CL-PD6729**Filename**

CL-PD6729.so

Classes

[CL-PD6729](#)

clipboard-gateway**Filename**

clipboard-gateway.so

Classes

[clipboard-gateway](#)

clock**Filename**

clock.so

Classes

[cell-and-clocks](#)
[clock](#)

Global Commands

[create-cell-and-clocks](#) create a non-instantiated cell-and-clocks
[new-cell-and-clocks](#) create an instantiated cell-and-clocks

cp3_quad100tx**Filename**

cp3_quad100tx.pymod

Classes

[cp3_quad100tx](#)

Global Commands

[create-cp3-quad100tx](#) create a non-instantiated cp3_quad100tx
[new-cp3-quad100tx](#) create an instantiated cp3_quad100tx

cpu-group**Filename**

cpu-group.so

Classes

[cpu-group](#)

cpumode-software-tracker**Filename**

cpumode_software_tracker.pymod

datagram-link**Filename**

datagram-link.so

Classes

[datagram_link](#)
[datagram_link_endpoint](#)
[datagram_link_impl](#)

Global Commands

[create-datagram-link](#) create a a non-instantiated datagram_link
[new-datagram-link](#) create a an instantiated datagram_link

DEC21041**Filename**

DEC21041.so

Classes

[DEC21041](#)

DEC21140A**Filename**

DEC21140A.so

Classes

[DEC21140A](#)

DEC21140A-dml**Filename**

DEC21140A-dml.so

Classes

[DEC21140A-dml](#)

DEC21143**Filename**

DEC21143.so

Classes

[DEC21143](#)

disassemble_v9**Filename**

disassemble_v9.so

Classes

[disassemble_v9](#)

disassemble_x86**Filename**

disassemble_x86.so

Classes

[disassemble_x86](#)

dm9161**Filename**

dm9161.so

Classes

[dm9161](#)

empty-device-c**Filename**

empty-device-c.so

Classes

[empty-device-c](#)

empty-device-cc**Filename**

empty-device-cc.so

Classes

[empty-device-cc](#)

etg**Filename**

etg.so

Classes[etg](#)**Global Commands**[new-etg](#) *deprecated* — Create an Ethernet traffic generator**eth-injector****Filename**`eth_injector.pymod`**Classes**[eth_injector](#)**eth-links****Filename**`eth-links.so`**Classes**[eth-cable-link](#)
[eth-cable-link-endpoint](#)
[eth-hub-link](#)
[eth-hub-link-endpoint](#)
[eth-link-snoop-endpoint](#)
[eth-switch-link](#)
[eth-switch-link-endpoint](#)
[eth-switch-link-snoop-endpoint](#)
[ethernet_cable](#)
[ethernet_hub](#)
[ethernet_switch](#)
[ethernet_vlan_switch](#)**Global Commands**

create-ethernet-cable	create a a non-instantiated ethernet_cable
create-ethernet-hub	create a a non-instantiated ethernet_hub
create-ethernet-switch	create a a non-instantiated ethernet_switch
create-ethernet-vlan-switch	create a a non-instantiated ethernet_vlan_switch
new-ethernet-cable	create a an instantiated ethernet_cable
new-ethernet-hub	create a an instantiated ethernet_hub
new-ethernet-switch	create a an instantiated ethernet_switch

new-ethernet-vlan-switch	create a an instantiated ethernet_vlan_switch
pcap-dump	Dump network traffic to a file, in libpcap format
pcap-dump-stop	stop the current dump
tcpdump	run the tcpdump program
tcpdump-stop	stop the current tcpdump capture
wireshark	run the wireshark/ethereal program
wireshark-stop	stop the current wireshark capture

eth-probe

Filename

`eth-probe.so`

Classes

[eth-probe](#)
[eth-probe-port](#)

Global Commands

[create-unconnected-ethernet-probe](#)
[insert-ethernet-probe](#)

eth-transceiver

Filename

`eth-transceiver.so`

Classes

[eth-transceiver](#)

eth_injector_comp

Filename

`eth_injector_comp.pymod`

Classes

[eth_injector_comp](#)

Global Commands

[create-eth-injector-comp](#) create a a non-instantiated eth_injector_comp
[new-eth-injector-comp](#) create a an instantiated eth_injector_comp

ethernet-link

Filename

ethernet-link.so

Classes

[ethernet-link](#)

Haps

[Ethernet_Frame](#)

fc-disk

Filename

fc-disk.so

Classes

[fc-disk](#)

Haps

[FC_SCSI_Command](#)

file-cdrom

Filename

file-cdrom.so

Classes

[file-cdrom](#)

Global Commands

[new-file-cdrom](#) create new file-cdrom object

firewire-bus

Filename

firewire-bus.so

Classes

[firewire_bus](#)

[firewire_device_test_wrapper](#)

Haps

[Firewire_Reset](#)
[Firewire_Transfer](#)

firewire-components

Filename

`firewire_components.pymod`

Classes

[pci-tsb12lv26](#)
[std-firewire-bus](#)
[std-firewire-sample-device](#)

Global Commands

[create-pci-tsb12lv26](#)
[create-std-firewire-bus](#)
[create-std-firewire-sample-device](#)
[new-std-firewire-bus](#)
[new-std-firewire-sample-device](#)

create a non-instantiated pci-tsb12lv26
create a non-instantiated std-firewire-bus
create a non-instantiated std-firewire-sample-device
create an instantiated std-firewire-bus
create an instantiated std-firewire-sample-device

firewire-sample-device

Filename

`firewire-sample-device.so`

Classes

[firewire_sample_device](#)

flash-memory

Filename

`flash-memory.so`

Classes

[flash-memory](#)

frequency-bus

Filename

`frequency-bus.so`

Classes

[frequency_bus](#)

frontend-server**Filename**

frontend-server.so

Classes

[frontend-server](#)

Global Commands

[new-remote-frontend](#) create a remote debugger session

frontend-server-console**Filename**

frontend-server-console.so

Classes

[frontend-server-console](#)

ftp-service**Filename**

ftp-service.so

Classes

[ftp-control](#)
[ftp-data](#)
[ftp-service](#)

ftp_alg**Filename**

ftp_alg.so

Classes

[ftp-alg](#)

g-cache

Filename

g-cache.so

Classes

[g-cache](#)
[trans-sorter](#)
[trans-splitter](#)

g-link

Filename

g-link.so

Classes

[generic-message-link](#)

Haps

[Generic_Message_Frame](#)

Global Commands

[new-glink](#) create a new generic message link

gdb-remote

Filename

gdb-remote.so

Classes

[gdb-remote](#)

Global Commands

[new-gdb-remote](#) Create a gdb session

generic-message-sample-device

Filename

generic-message-sample-device.so

Classes

[generic-message-sample-device](#)

generic mmc card**Filename**

generic mmc card.so

Classes

[generic mmc card](#)

generic pcie switch comp**Filename**

generic_pcie_switch_comp.pymod

Classes

[generic_pcie_switch](#)

Global Commands

[create-generic-pcie-switch](#) create a a non-instantiated generic_pcie_switch
[new-generic-pcie-switch](#) create a an instantiated generic_pcie_switch

generic pcie switch port**Filename**

generic_pcie_switch_port.so

Classes

[generic_pcie_switch_port](#)

generic spi flash**Filename**

generic_spi_flash.so

Classes

[generic_spi_flash](#)

generic eth phy

Filename

generic_eth_phy.so

Classes

[generic_eth_phy](#)

hap-meter**Filename**

hap-meter.so

Classes

[hap-meter](#)

Global Commands

[new-hap-meter](#) create a new hap meter

host-condor-pci-1553**Filename**

host-condor-pci-1553.so

Classes

[host-condor-pci-1553](#)

host-pmc-1553**Filename**

host-pmc-1553.so

Classes

[host-pmc-1553](#)

host-serial-console**Filename**

host-serial-console.so

Classes

[host-serial-console](#)

hostfs

Filename

hostfs.so

Classes

[hostfs](#)

hypersim-pattern-matcher

Filename

hypersim-pattern-matcher.so

Classes

[hypersim-pattern-matcher](#)

Global Commands

[disable-hypersim](#)

[enable-hypersim](#)

[hypersim-status](#)

[list-hypersim-patterns](#)

enables/disables hypersimulation

show hypersim status

list available hypersim patterns

i21150

Filename

i21150.so

Classes

[i21150](#)

i21152

Filename

i21152.so

Classes

[i21152](#)

i21154

Filename

i21154.so

Classes

[i21154](#)

i21554

Filename

i21554.so

Classes

[i21554-prim](#)
[i21554-scnd](#)

i21555

Filename

i21555.so

Classes

[i21555-prim](#)
[i21555-scnd](#)

i2c-bus

Filename

i2c-bus.so

Classes

[i2c-bus](#)
[i2c_slave_v2_to_bus_adapter](#)

i2c-link-v1

Filename

i2c-link-v1.so

Classes

[i2c_link_v1](#)

i2c-link-v2

Filename

i2c-link-v2.so

Classes

[i2c-link-endpoint](#)
[i2c-link-impl](#)
[i2c_link_v2](#)
[i2c_wire](#)

Global Commands

[create-i2c-link-v2](#) create a a non-instantiated i2c_link_v2
[new-i2c-link-v2](#) create a an instantiated i2c_link_v2

i82543

Filename

i82543.so

Classes

[i82543](#)

i82546

Filename

i82546.so

Classes

[i82546](#)

i82559

Filename

i82559.so

Classes

[i82559](#)

id-splitter

Filename

[id-splitter.so](#)

Classes

[id-splitter](#)

ide**Filename**

[ide.so](#)

Classes

[ide](#)
[ide-cdrom](#)
[ide-disk](#)

isa-components**Filename**

[isa_components.pymod](#)

Classes

[isa-fourport](#)
[isa-lance](#)
[isa-vga](#)
[pc-dual-serial-ports](#)
[pc-floppy-controller](#)
[pc-quad-serial-ports](#)
[pc-single-parallel-port](#)
[ps2-keyboard-mouse](#)
[sio-w83627hf](#)
[std-super-io](#)

Global Commands

[create-isa-fourport](#)
[create-isa-lance](#)
[create-isa-vga](#)
[create-pc-dual-serial-ports](#)
[create-pc-floppy-controller](#)
[create-pc-quad-serial-ports](#)
[create-pc-single-parallel-port](#)
[create-ps2-keyboard-mouse](#)
[create-sio-w83627hf](#)
[create-std-super-io](#)

create a non-instantiated isa-fourport
create a non-instantiated isa-lance
create a non-instantiated isa-vga
create a non-instantiated pc-dual-serial-ports
create a non-instantiated pc-floppy-controller
create a non-instantiated pc-quad-serial-ports
create a non-instantiated pc-single-parallel-port
create a non-instantiated ps2-keyboard-mouse
create a non-instantiated sio-w83627hf
create a non-instantiated std-super-io

ISP1040

Filename

ISP1040.so

Classes

[ISP1040](#)

ISP2200

Filename

ISP2200.so

Classes

[ISP2200](#)

mac-splitter

Filename

mac-splitter.so

Classes

[mac_splitter](#)

mem-traffic-meter

Filename

mem-traffic-meter.so

Classes

[mem-traffic-meter](#)

Global Commands

[new-mem-traffic-meter](#) create a new memory traffic meter

memory-comp

Filename

memory_comp.pymod

Classes

[simple_memory_module](#)

Global Commands

[create-simple-memory-module](#)
[new-simple-memory-module](#)

create a a non-instantiated simple_memory_module
create a an instantiated simple_memory_module

memory-components**Filename**

`memory_components.pymod`

Classes

[ddr-memory-module](#)
[ddr2-memory-module](#)
[ddr3-memory-module](#)
[sdram-memory-module](#)

Global Commands

[create-and-connect-ddr-memory](#)
[create-ddr-memory-module](#)
[create-ddr2-memory-module](#)
[create-ddr3-memory-module](#)
[create-sdram-memory-module](#)

create and connect memory modules to the system
create a non-instantiated ddr-memory-module
create a non-instantiated ddr2-memory-module
create a non-instantiated ddr3-memory-module
create a non-instantiated sdram-memory-module

microwire-eeprom**Filename**

`microwire-eeprom.so`

Classes

[microwire-eeprom](#)

mii-management-bus**Filename**

`mii-management-bus.so`

Classes

[mii-management-bus](#)

mii-transceiver

Filename

mii-transceiver.so

Classes

[mii-transceiver](#)

mil-std-1553-components

Filename

mil_std_1553_components.pymod

Classes

[pci-pmc1553](#)
[std-ms1553-link](#)

Global Commands

create-pci-pmc1553	create a non-instantiated pci-pmc1553
create-std-ms1553-link	create a non-instantiated std-ms1553-link
new-std-ms1553-link	create an instantiated std-ms1553-link

mmc-card-components

Filename

mmc_card_comp.pymod

Classes

[micron_mtfc2ggqdi_emmc_card](#)
[micron_mtfc4ggqdi_emmc_card](#)
[micron_mtfc4ggqdi_sdhc_card](#)

Global Commands

create-micron-mtfc2ggqdi-emmc-card	create a a non-instantiated micron_mtfc2ggqdi_emmc_card
create-micron-mtfc4ggqdi-emmc-card	create a a non-instantiated micron_mtfc4ggqdi_emmc_card
create-micron-mtfc4ggqdi-sdhc-card	create a a non-instantiated micron_mtfc4ggqdi_sdhc_card
new-micron-mtfc2ggqdi-emmc-card	create a an instantiated micron_mtfc2ggqdi_emmc_card
new-micron-mtfc4ggqdi-emmc-card	create a an instantiated micron_mtfc4ggqdi_emmc_card
new-micron-mtfc4ggqdi-sdhc-card	create a an instantiated micron_mtfc4ggqdi_sdhc_card

ms1553-link

Filename

ms1553-link.so

Classes

[ms1553-link](#)
[ms1553-recorder-bm](#)

ms1553-rt**Filename**

ms1553-rt.so

Classes

[ms1553_rt](#)

ms1553-test-rt**Filename**

ms1553_test_rt.pymod

Classes

[ms1553-test-rt](#)

mtprof**Filename**

mtprof.so

Classes

[mtprof](#)

Global Commands

[disable-mtprof](#) disable mtprof data collection
[enable-mtprof](#) enable multithreaded simulation profiling

new-flash-memory**Filename**

new-flash-memory.so

Classes

[generic-flash-memory](#)

NS16550_c

Filename

NS16550_c.so

Classes

[NS16550_c](#)

NS16x50

Filename

NS16x50.so

Classes

[NS16450](#)

[NS16550](#)

onfi-flash

Filename

onfi-flash.so

Classes

[onfi_flash](#)

os-awareness

Filename

os_awareness.pymod

Classes

[coverage_profiler](#)

[os_awareness](#)

Global Commands

[create-os-awareness](#) create a a non-instantiated os_awareness

[new-os-awareness](#) create a an instantiated os_awareness

PCF8582C

Filename

PCF8582C.so

Classes[PCF8582C](#)**pci-bus****Filename**

pci-bus.so

Classes[pci-bus](#)**pci-components****Filename**

pci_components.pymod

Classes

- [pci-accel-vga](#)
- [pci-am79c973](#)
- [pci-bcm5703c](#)
- [pci-bcm5704c](#)
- [pci-dec21041](#)
- [pci-dec21140a](#)
- [pci-dec21140a-dml](#)
- [pci-dec21143](#)
- [pci-i21152](#)
- [pci-i82543gc](#)
- [pci-i82546bg](#)
- [pci-i82559](#)
- [pci-ispl040](#)
- [pci-ispl2200](#)
- [pci-pd6729](#)
- [pci-sil680a](#)
- [pci-sym53c810](#)
- [pci-sym53c875](#)
- [pci-sym53c876](#)
- [pci-vga](#)

Global Commands[create-pci-accel-vga](#)

create a non-instantiated pci-accel-vga

[create-pci-am79c973](#)

create a non-instantiated pci-am79c973

[create-pci-bcm5703c](#)

create a non-instantiated pci-bcm5703c

create-pci-bcm5704c	create a non-instantiated pci-bcm5704c
create-pci-dec21041	create a non-instantiated pci-dec21041
create-pci-dec21140a	create a non-instantiated pci-dec21140a
create-pci-dec21140a-dml	create a non-instantiated pci-dec21140a-dml
create-pci-dec21143	create a non-instantiated pci-dec21143
create-pci-i21152	create a non-instantiated pci-i21152
create-pci-i82543gc	create a non-instantiated pci-i82543gc
create-pci-i82546bg	create a non-instantiated pci-i82546bg
create-pci-i82559	create a non-instantiated pci-i82559
create-pci-isp1040	create a non-instantiated pci-isp1040
create-pci-isp2200	create a non-instantiated pci-isp2200
create-pci-pd6729	create a non-instantiated pci-pd6729
create-pci-sil680a	create a non-instantiated pci-sil680a
create-pci-sym53c810	create a non-instantiated pci-sym53c810
create-pci-sym53c875	create a non-instantiated pci-sym53c875
create-pci-sym53c876	create a non-instantiated pci-sym53c876
create-pci-vga	create a non-instantiated pci-vga

pcie-bus

Filename

`pcie-bus.so`

Classes

[pcie-bus](#)

phy-comp

Filename

`phy_comp.pymod`

Classes

[phy_comp](#)

Global Commands

create-phy-comp	create a a non-instantiated phy_comp
new-phy-comp	create a an instantiated phy_comp

phy-components

Filename

`phy_components.pymod`

Classes

[phy-mii-transceiver](#)

Global Commands

[create-phy-mii-transceiver](#) create a non-instantiated phy-mii-transceiver

PMC1553**Filename**

PMC1553.so

Classes

[PMC1553](#)

rapidio-link**Filename**

rapidio-link.so

Classes

[rapidio_link](#)
[rapidio_link_endpoint](#)
[rapidio_link_impl](#)

Global Commands

[create-rapidio-link](#) create a a non-instantiated rapidio_link

[new-rapidio-link](#) create a an instantiated rapidio_link

rapidio-simple-device**Filename**

rapidio-simple-device.so

Classes

[rapidio_simple_device](#)

rapidio-tape**Filename**

rapidio_tape.pymod

Classes

[rapdio_tape](#)

Global Commands

[new-rapdio-tape](#) start traffic generator

real-network

Filename

real-network.so

Classes

[real-network-bridge](#)
[real-network-host](#)
[real-network-router](#)
[rn-eth-bridge-raw](#)
[rn-eth-bridge-tap](#)
[rn-eth-proxy-raw](#)
[rn-ip-router-raw](#)

Global Commands

[close-tap-interface](#)
[connect-real-network-bridge](#)
[connect-real-network-host](#)
[connect-real-network-router](#)
[create-real-network-bridge](#)
[create-real-network-host](#)
[create-real-network-router](#)
[disconnect-real-network](#)
[network-helper](#)
[new-real-network-bridge](#)
[new-real-network-host](#)
[new-real-network-router](#)

close an unused persistent TAP interface
connect bridge between real and simulated network
connect real host to the simulated network
deprecated — connect router between real and simulated network
create a non-instantiated real-network-bridge
create a non-instantiated real-network-host
create a non-instantiated real-network-router
disconnect from the real network
set/show name of host network helper
create an instantiated real-network-bridge
create an instantiated real-network-host
create an instantiated real-network-router

realtime

Filename

realtime.so

Classes

[realtime](#)

Global Commands

disable-real-time-mode	disable real-time behavior
enable-real-time-mode	enable real-time behavior
new-realtime	create a new realtime object

recorder**Filename**

`recorder.so`

Classes

[recorder](#)

scsi-bus**Filename**

`scsi-bus.so`

Classes

[scsi-bus](#)

scsi-cdrom**Filename**

`scsi-cdrom.so`

Classes

[scsi-cdrom](#)

Haps

[SCSI_CDROM_Command](#)

scsi-disk**Filename**

`scsi-disk.so`

Classes

[scsi-disk](#)

Haps

[SCSI_Disk_Command](#)

selfprof**Filename**

selfprof.so

Classes

[selfprof](#)

ser-link**Filename**

ser-link.so

Classes

[ser-link-endpoint](#)
[ser-link-impl](#)
[ser_link](#)

Global Commands

[create-ser-link](#) create a a non-instantiated ser_link
[new-ser-link](#) create a an instantiated ser_link

serial-link**Filename**

serial-link.so

Classes

[serial-link](#)

service-node**Filename**

service-node.so

Classes

[port-forward-ingress-server](#)
[port-forward-egress-server](#)
[service-node](#)
[service-node-device](#)
[std-service-node](#)

Global Commands

connect-real-network	connect a simulated machine to the real network
connect-real-network-napt	enable NAPT from simulated network
connect-real-network-port-in	setup port forwarding to a simulated machine
connect-real-network-port-out	setup port forwarding to real machine
create-std-service-node	create a non-instantiated std-service-node
default-port-forward-target	set default port forwarding target
disconnect-real-network-port-in	remove port forwarding to a simulated machine
disconnect-real-network-port-out	remove port forwarding to real machine
list-port-forwarding-setup	view the port forwarding setup
new-std-service-node	create an instantiated std-service-node

set-memory**Filename**

set-memory.so

Classes

[set-memory](#)

signal-bus**Filename**

signal-bus.so

Classes

[signal-bus](#)

signal-link**Filename**

signal-link.so

Classes

[signal_link](#)
[signal_link_endpoint](#)
[signal_link_impl](#)

Global Commands

create-signal-link	create a a non-instantiated signal_link
new-signal-link	create a an instantiated signal_link

signal-to-interrupt

Filename

signal-to-interrupt.so

Classes

[signal_to_interrupt](#)

sil680a

Filename

sil680a.so

Classes

[SIL680A](#)

Simics Core

Filename

libsimics-common.so

Classes

[branch_recorder](#)
[breakpoints](#)
[cell](#)
[cli](#)
[component](#)
[connector](#)
[context](#)
[data-profiler](#)
[dynamic_link_connector](#)
[fake-space](#)
[gui](#)
[image](#)
[memory-space](#)
[perfanalyze-client](#)
[persistent-ram](#)
[port-space](#)
[preferences](#)
[ram](#)
[remote_sync_domain](#)

remote_sync_node
remote_sync_server
rev-execution
rom
sim
slaver
static_link_connector
sync_domain
terminal_frontend
top-component

Haps

CLI_Command_Added
CLI_Variable_Write
Component_Change
Component_Hierarchy_Change
Core_Address_Not_Mapped
Core_Asynchronous_Trap
Core_At_Exit
Core_Back_To_Front
Core_Breakpoint_Change
Core_Breakpoint_Memop
Core_Conf_Class_Register
Core_Conf_Class_Unregister
Core_Conf_Clock_Change_Cell
Core_Conf_Object_Change_Clock
Core_Conf_Object_Create
Core_Conf_Object_Created
Core_Conf_Object_Delete
Core_Conf_Object_Pre_Delete
Core_Conf_Object_Rename
Core_Conf_Objects_Created
Core_Conf_Objects_Deleted
Core_Configuration_Loaded
Core_Context_Activate
Core_Context_Change
Core_Context_Deactivate
Core_Context_Updated
Core_Continuation
Core_Control_Register_Read
Core_Control_Register_Write
Core_Cycle_Count

Core_DSTC_Flush_Counter
Core_Device_Access_Memop
Core_Disable_Breakpoints
Core_Discard_Future
Core_Exception
Core_Exception_Return
Core_External_Interrupt
Core_Frequency_Changed
Core_Global_Message
Core_Hap_Callback_Installed
Core_Hap_Callback_Removed
Core_Hap_Type_Added
Core_Image_Activity
Core_Initial_Configuration
Core_Log_Message
Core_Log_Message_Extended
Core_Magic_Instruction
Core_Memory_Space_Map_Changed
Core_Mode_Change
Core_Module_Loaded
Core_Multithreading_Changed
Core_NotImplemented
Core_Periodic_Event
Core_Preferences_Changed
Core_Processor_Schedule_Changed
Core_Recent_Files_Changed
Core_Rexec_Active
Core_Simulation_Mode_Change
Core_Simulation_Stopped
Core_Skipto_Progress
Core_Source_Step
Core_Step_Count
Core_Sync_Instruction
Core_Time_Transition
Core_Timing_Model_Change
Core_Workspace_Changed
Core_Write_Configuration
Internal_Bookmark_List_Changed
Internal_Micro_Checkpoint_Loaded
Internal_SB_Barrier
Internal_SB_Command_File

[Internal_SB_Cycle_Count](#)
[Internal_SB_Log](#)
[Internal_SB_Pipe](#)
[Internal_SB_Step_Count](#)
[Internal_SB_Time](#)
[Internal_SB_Variable](#)
[Internal_Time_Direction_Changed](#)
[Internal_Time_Quantum_Changed](#)
[Rexec_Limit_Exceeded](#)
[UI_Run_State_Changed](#)

Global Commands

!	execute a shell command
!=	not equal
#	treat the line as a comment
\$	get the value of a CLI variable
%	read register by name, module or string formatting
&	bitwise AND operation
*	arithmetic multiplication
+	arithmetic addition, string and list concatenation
+=	set a CLI variable
-	arithmetic subtraction
-=	set a CLI variable
->	access object attribute
/	arithmetic division
:	access component object in a component
<	less than
<<	bitwise left shift
<=	less or equal
=	set a CLI variable
==	equal
>	greater than
>=	greater or equal
>>	bitwise right shift
@	evaluate a Python statement
[
^	bitwise XOR operation
'	evaluate a Python expression
add-data-to-script-pipe	send data to a script pipe
add-directory	add a directory to the Simics search path
add-module-directory	add a directory to the module search path
alias	add an alias

and	logical and
api-help	get API help
api-search	search API help
auto-partition-configuration	suggest a cell partitioning
bin	display integer in binary notation
break	set breakpoint on current processor
break-cr	break on control register updates
break-exception	break on CPU exceptions
break-hap	break on haps
break-io	break on device accesses
break-log	break on log message
cd	change working directory
change-namespace	change current namespace
check-cell-partitioning	verify that cell partitioning is OK
clear-directories	clear the Simics search path
clear-recorder	clear recorded events
command-history	show CLI command history
command-list	generate html document describing commands
configuration-shortest-paths	find shortest paths between two objects
connect	connect connectors
connect-components	<i>deprecated</i> — connect components
copy-connector	copy object
copyright	print full Simics copyright information
cpu-switch-time	get/set default switch time
create-script-barrier	create a script barrier
create-script-pipe	create a script pipe
current-namespace	return current namespace
current-processor	return current processor
cycle-break	set cycle breakpoint
cycle-break-absolute	set absolute cycle breakpoint
date	host time and date
dec	display integer in decimal notation
defined	variable defined
delete	remove breakpoints
delete-bookmark	delete a time bookmark
devs	list all devices in Simics
digit-grouping	set output formatting for numbers
dirs	display directory stack
disable	enable/disable breakpoint
disable-magic-breakpoint	remove breakpoint on magic break instructions
disable-multithreading	disable multithreading

disable-page-sharing	disable page-sharing
disable-reverse-execution	disable reverse execution
disassemble	disassemble instructions
disassemble-settings	change disassembly output settings
disconnect	disconnect connectors
display	print expression at prompt
down	go down N stack frames
dstc-disable	disable D-STC
dstc-enable	enable D-STC
echo	echo a value to screen
else	
enable	enable/disable breakpoint
enable-magic-breakpoint	set breakpoint on magic break instructions
enable-multithreading	enable multithreading
enable-page-sharing	enable page-sharing
enable-reverse-execution	enable reverse execution
env	
except	
exec	execute a string as a CLI command
expect	fail if not equal
file-exists	check if a file exists in the search path
finish-function	finish the current function
foreach	
frame	change current stack frame
get	get value of physical address
get-breakpoint-list	return list of all breakpoints
get-class-list	return a list all configuration classes
get-component-list	return component list
get-component-prefix	<i>deprecated</i> — get current component name prefix
get-error-command	return the name of command causing error
get-error-file	return the file name of the CLI command error
get-error-line	return the file line number of the CLI command error
get-error-message	return the message for an error
get-object-list	return a list of all objects
help	help command
help-search	search for text in documentation
hex	display integer in hexadecimal notation
if	
ignore	set ignore count for a breakpoint
in	check for occurrence in string or list
in-list	check for occurrence in list

instantiate-components	instantiate components
instruction-fetch-mode	set or get current mode for instruction fetching
int-to-double-float	interpret integer as 64-bit floating point
int-to-extended-double-float	interpret integer as 80-bit floating point
int-to-quad-float	interpret integer as 128-bit floating point
int-to-single-float	interpret integer as 32-bit floating point
interrupt-script	interrupt script execution
interrupt-script-branch	interrupt the execution of a script branch
io-stats	list most frequently accessed devices
iostc-disable	disable IO-STC
iostc-enable	enable IO-STC
istc-disable	disable I-STC
istc-enable	enable I-STC
license	print Simics license
list	list source and/or disassemble
list-attributes	list all attributes
list-bookmarks	list time bookmarks
list-breakpoints	print information about breakpoints
list-checkpoints	list checkpoints
list-classes	list all configuration classes
list-components	list components
list-directories	list directories in Simics search path
list-failed-modules	list the modules that are not loadable
list-hap-callbacks	print list of hap callbacks
list-haps	print list of haps
list-length	returns the length of a list
list-modules	list loadable modules
list-namespaces	list all namespaces
list-objects	list objects
list-objects-with-interface	Lists configuration objects implementing specific interface.
list-preferences	list preference
list-script-branches	list all script branches
list-variables	list CLI variables
load-binary	load binary (executable) file into memory
load-file	load file into memory
load-module	load module into Simics
load-persistent-state	load persistent state
local	define a local variable
log	print log entries for all objects
log-level	set or get the global log level
log-setup	configure log behavior

<code>log-size</code>	set log buffer size
<code>log-type</code>	set or get the current log types
<code>logical-to-physical</code>	translate logical address to physical
<code>ls</code>	list files
<code>magic-breakpoint-enabled</code>	return TRUE if the magic breakpoint is enabled
<code>match-string</code>	compare string with a pattern and return matches
<code>max</code>	max
<code>min</code>	min
<code>module-list-refresh</code>	create a new list of loadable modules
<code>move-object</code>	move object
<code>native-path</code>	convert a filename to host native form
<code>new-context</code>	create a new context
<code>next-instruction</code>	run to the next instruction, skipping subroutine calls
<code>next-line</code>	run to the next source line, skipping subroutine calls
<code>not</code>	logical not
<code>object-exists</code>	check if object exists
<code>oct</code>	display integer in octal notation
<code>or</code>	logical or
<code>output-file-start</code>	send output to file
<code>output-file-stop</code>	stop sending output to file
<code>output-radix</code>	change the default output radix
<code>pdisable</code>	switch processor off
<code>penable</code>	switch processor on
<code>pid</code>	print pid of Simics process
<code>pipe</code>	run commands through a pipe
<code>popd</code>	pop directory from directory stack
<code>pos</code>	address of line or function
<code>pow</code>	power of
<code>pregs</code>	print cpu registers
<code>print</code>	display integer in various bases
<code>print-event-queue</code>	print event queue for processor
<code>print-time</code>	print number of steps and cycles executed
<code>pselect</code>	select a processor
<code>pstatus</code>	show processors' status
<code>psym</code>	print value of symbolic expression
<code>pushd</code>	push directory on directory stack
<code>pwd</code>	print working directory
<code>python</code>	evaluate a Python expression
<code>quit</code>	quit from Simics
<code>range</code>	create and return a list of integers
<code>read-configuration</code>	restore configuration

<code>read-reg</code>	read a register
<code>read-variable</code>	value of a named variable
<code>readme</code>	print information about Simics
<code>resolve-file</code>	resolve a filename
<code>restart-simics</code>	restart the current simics session
<code>reverse</code>	run simulation backwards
<code>reverse-cycles</code>	run simulation backwards
<code>reverse-next-instruction</code>	reverse to the previous instruction, skipping subroutine calls
<code>reverse-next-line</code>	reverse to the previous source line, skipping subroutine calls
<code>reverse-step-instruction</code>	reverse step one or more instruction
<code>reverse-step-line</code>	reverse to the previous source line
<code>reverse-to</code>	set a temporary time breakpoint and run backwards
<code>rexec-limit</code>	tune reverse execution performance parameters
<code>run</code>	start execution
<code>run-command-file</code>	execute a simics script
<code>run-cycles</code>	start execution
<code>run-python-file</code>	execute Python file
<code>run-seconds</code>	execute for seconds
<code>save-component-template</code>	save a component configuration template
<code>save-persistent-state</code>	save persistent simulator state
<code>save-preferences</code>	save preference
<code>script-branch</code>	check if script pipe contains data
<code>script-pipe-has-data</code>	set physical address to specified value
<code>set</code>	set a bookmark at the current point in time
<code>set-bookmark</code>	<i>deprecated</i> — set a prefix for all component names
<code>set-component-prefix</code>	set the current context of a CPU
<code>set-context</code>	limit memory usage
<code>set-memory-limit</code>	set the min latency of the sync domain
<code>set-min-latency</code>	set an instruction pattern for a breakpoint
<code>set-pattern</code>	set the current processor's program counter
<code>set-pc</code>	set a syntax prefix for a breakpoint
<code>set-prefix</code>	set a syntax substring for a breakpoint
<code>set-substr</code>	limit the number of simulation threads
<code>set-thread-limit</code>	execute a shell command
<code>shell</code>	interpret unsigned integer as signed
<code>signed</code>	interpret unsigned integer as signed
<code>signed16</code>	interpret unsigned integer as signed
<code>signed32</code>	interpret unsigned integer as signed
<code>signed64</code>	interpret unsigned integer as signed
<code>signed8</code>	interpret unsigned integer as signed
<code>simulation-replaying</code>	check if simulation is replaying

<code>simulation-reversing</code>	check if simulation is reversing
<code>simulation-running</code>	check if simulation is running
<code>skip-to</code>	skip to the specified point in the simulation
<code>split-string</code>	split string based on its type
<code>stack-trace</code>	display stack trace
<code>stc-status</code>	show I- and D-STC status
<code>step-break</code>	set time breakpoint
<code>step-break-absolute</code>	set absolute time breakpoint
<code>step-cycle</code>	step one or more cycles
<code>step-instruction</code>	step one or more instructions
<code>step-line</code>	run to the next source line
<code>stop</code>	interrupt simulation
<code>sym-address</code>	return the address of expression or source line
<code>sym-file</code>	return source file for function or address
<code>sym-function</code>	return function at a given address
<code>sym-line</code>	return source line for function or address
<code>sym-source</code>	print source location for function or address
<code>sym-string</code>	evaluate symbolic expression
<code>sym-type</code>	return the type of a symbolic expression
<code>sym-value</code>	evaluate symbolic expression
<code>symval</code>	evaluate symbolic expression
<code>sync-info</code>	Print synchronization configuration
<code>system-info</code>	system info
<code>trace-breakpoint</code>	trace breakpoints
<code>trace-cr</code>	trace control register updates
<code>trace-exception</code>	trace exceptions
<code>trace-hap</code>	trace haps
<code>trace-io</code>	trace device accesses
<code>trace-io-setup</code>	
<code>try</code>	runs a block of commands and catches any error
<code>unbreak</code>	remove breakpoint range
<code>unbreak-cr</code>	break on control register updates
<code>unbreak-exception</code>	break on CPU exceptions
<code>unbreak-hap</code>	break on haps
<code>unbreak-io</code>	break on device accesses
<code>uncall-function</code>	reverse to when the current function was called
<code>undisplay</code>	remove expression installed by display
<code>unload-module</code>	unload module
<code>unset</code>	remove a CLI variable
<code>untrace-breakpoint</code>	trace breakpoints
<code>untrace-cr</code>	trace control register updates

untrace-exception	trace exceptions
untrace-hap	trace haps
untrace-io	trace device accesses
up	go up N stack frames
version	display Simics version
wait-for-breakpoint	wait for a memory breakpoint to trigger
wait-for-hap	<i>deprecated</i> — wait until hap occurs
wait-for-script-barrier	wait until enough branches have reached a barrier
wait-for-script-pipe	wait until there is data on a script pipe
wait-for-variable	<i>deprecated</i> — wait for a variable to change
whereis	find symbol by address
while	
win-about	information on GUI load failure
write-configuration	save configuration
write-reg	write to register
x	examine raw memory contents
 	bitwise OR operation
~	bitwise not

simple-scsi-disk

Filename

[simple-scsi-disk.so](#)

Classes

[simple-scsi-disk](#)

Haps

[SCSI_Disk_Command](#)

software-tracker

Filename

[software-tracker.pymod](#)

Classes

[software_tracker](#)

software-tracker-iface

Filename

[software-tracker-iface.so](#)

state-assertion

Filename

state-assertion.so

Classes

[state-assertion](#)

Global Commands

[state-assertion-connect](#)
[state-assertion-create-file](#)
[state-assertion-open-file](#)
[state-assertion-receive](#)
[state-assertion-simple-assert](#)
[state-assertion-simple-record](#)

connect to a state-assertion receiver
record a state assertion file
open a state assertion file for comparing
wait for a connection from a state assertion sender
assert the file
record the state of an object every x steps

status-panel

Filename

status_panel.pymod

Classes

[status_panel](#)

status-panel-view

Filename

status_panel_view.pymod

Classes

[status_panel_view](#)

std-components

Filename

std_components.pymod

Classes

[dummy-component](#)
[etg_comp](#)
[etg_panel](#)

[example-keypad](#)
[example-status-panel](#)
[simple-fc-disk](#)
[std-etg](#)
[std-ethernet-link](#)
[std-generic-link](#)
[std-generic-link-sample](#)
[std-graphics-console](#)
[std-host-serial-console](#)
[std-ide-cdrom](#)
[std-ide-disk](#)
[std-pcmcia-flash-disk](#)
[std-scsi-bus](#)
[std-scsi-cdrom](#)
[std-scsi-disk](#)
[std-serial-link](#)
[std-server-console](#)
[std-telnet-console](#)
[std-text-console](#)
[std-text-graphics-console](#)
[std_mmc_card](#)
[std_sata_cdrom](#)
[std_sata_disk](#)

Global Commands

[create-dummy-component](#) create a non-instantiated dummy-component
[create-etg-comp](#) create a a non-instantiated etg_comp
[create-etg-panel](#) create a a non-instantiated etg_panel
[create-example-keypad](#) create a non-instantiated example-keypad
[create-example-status-panel](#) create a non-instantiated example-status-panel
[create-simple-fc-disk](#) create a non-instantiated simple-fc-disk
[create-std-etg](#) create a non-instantiated std-etg
[create-std-ethernet-link](#) create a non-instantiated std-ethernet-link
[create-std-generic-link](#) create a non-instantiated std-generic-link
[create-std-generic-link-sample](#) create a non-instantiated std-generic-link-sample
[create-std-graphics-console](#) create a non-instantiated std-graphics-console
[create-std-host-serial-console](#) create a non-instantiated std-host-serial-console
[create-std-ide-cdrom](#) create a non-instantiated std-ide-cdrom
[create-std-ide-disk](#) create a non-instantiated std-ide-disk
[create-std-mmc-card](#) create a a non-instantiated std_mmc_card
[create-std-pcmcia-flash-disk](#) create a non-instantiated std-pcmcia-flash-disk
[create-std-sata-cdrom](#) create a a non-instantiated std_sata_cdrom
[create-std-sata-disk](#) create a a non-instantiated std_sata_disk

create-std-scsi-bus	create a non-instantiated std-scsi-bus
create-std-scsi-cdrom	create a non-instantiated std-scsi-cdrom
create-std-scsi-disk	create a non-instantiated std-scsi-disk
create-std-serial-link	create a non-instantiated std-serial-link
create-std-server-console	create a non-instantiated std-server-console
create-std-telnet-console	create a non-instantiated std-telnet-console
create-std-text-console	create a non-instantiated std-text-console
create-std-text-graphics-console	create a non-instantiated std-text-graphics-console
new-dummy-component	create an instantiated dummy-component
new-etg-comp	create a an instantiated etg_comp
new-etg-panel	create a an instantiated etg_panel
new-example-keypad	create an instantiated example-keypad
new-example-status-panel	create an instantiated example-status-panel
new-std-etg	create an instantiated std-etg
new-std-ethernet-link	create an instantiated std-ethernet-link
new-std-generic-link	create an instantiated std-generic-link
new-std-graphics-console	create an instantiated std-graphics-console
new-std-host-serial-console	create an instantiated std-host-serial-console
new-std mmc-card	create a an instantiated std_mmc_card
new-std-pcmcia-flash-disk	create an instantiated std-pcmcia-flash-disk
new-std-sata-cdrom	create a an instantiated std_sata_cdrom
new-std-sata-disk	create a an instantiated std_sata_disk
new-std-scsi-bus	create an instantiated std-scsi-bus
new-std-serial-link	create an instantiated std-serial-link
new-std-server-console	create an instantiated std-server-console
new-std-telnet-console	create an instantiated std-telnet-console
new-std-text-console	create an instantiated std-text-console
new-std-text-graphics-console	create an instantiated std-text-graphics-console

SYM53C810

Filename

SYM53C810.so

Classes

[SYM53C810](#)

SYM53C875

Filename

SYM53C875.so

Classes

[SYM53C875](#)

symtable

Filename

`symtable.so`

Classes

`symtable`

Haps

`Symtable_Updated`

Global Commands

`exec-info`

`exec-info-clean`

`function-profile`

list functions sorted by profile counts

`new-symtable`

create new symbol table

`select-profiles`

set profilers to display in source listing

`turbo-disable-block-profile`

`turbo-enable-block-profile`

`turbo-info`

`turbo-info-clean`

`turbo-show-block-profile`

system-panel

Filename

`system-panel.so`

Classes

`system_panel_bool_in`

`system_panel_bool_out`

`system_panel_number_in`

`system_panel_number_out`

`system_panel_state_manager`

Global Commands

`connect-panel-to-frontend`

system-panel-interface

Filename

`system-panel-interface.so`

system-panel-text

Filename

system_panel_text.pymod

Classes

[text_panel_frontend](#)

system-perfmeter

Filename

system_perfmeter.pymod

Global Commands

system-perfmeter	activate Simics performance monitoring
system-perfmeter-plot	create graphs from performance data

tcf-agent

Filename

tcf-agent.so

Classes

[tcf-agent](#)
[tcf-context-proxy](#)

Global Commands

add-pathmap-entry	add a path map entry
add-symbol-file	add symbol file to debugging contexts
break-line	Add breakpoint at a source code line
break-location	Add breakpoint at a location
clear-memorymap	clear all memory map entries
clear-pathmap	clear all path map entries
debug-context	return the current debug object
list-debug-contexts	List debug contexts
list-sections	Lists the relocatable sections of a symbol file
list-segments	Lists the segments of a symbol file
new-tcf-agent	create a tcf agent
show-memorymap	show the current memory map
show-pathmap	show the current path map

telnet-console

Filename

telnet-console.so

Classes

[telnet_console](#)

Haps

[Xterm_Break_String](#)

telnet-frontend**Filename**

telnet-frontend.so

Classes

[telnet_frontend](#)

Global Commands

[telnet-frontend](#) activate telnet based command line

time-server-c**Filename**

time-server-c.so

Classes

[time-server](#)

Global Commands

[new-time-server](#) Create a new time server

timing-components**Filename**

timing_components.pymod

Classes

[instruction-data-splitter](#)
[memory-timer](#)
[sample-gcache](#)

Global Commands

<code>create-instruction-data-splitter</code>	create a non-instantiated instruction-data-splitter
<code>create-memory-timer</code>	create a non-instantiated memory-timer
<code>create-sample-gcache</code>	create a non-instantiated sample-gcache
<code>new-sample-gcache</code>	create an instantiated sample-gcache

trace**Filename**`trace.so`**Classes**

<code>base-trace-mem-hier</code>
<code>trace-mem-hier</code>

Global Commands

<code>new-tracer</code>	create a new tracer
-------------------------	---------------------

trace-sync**Filename**`trace-sync.so`**Classes**

<code>trace-sync</code>

Global Commands

<code>enable-core2-bugfix</code>	enable hardware bug workaround
----------------------------------	--------------------------------

trans-staller**Filename**`trans-staller.so`**Classes**

<code>trans-staller</code>

TSB12LV26**Filename**`TSB12LV26.so`

Classes[TSB12LV26](#)**tsi500****Filename**

tsi500.so

Classes[tsi500](#)**usb-comp****Filename**

usb_comp.pymod

Classes[usb_xmas_tree](#)**Global Commands**[create-usb-xmas-tree](#) create a non-instantiated usb_xmas_tree[new-usb-xmas-tree](#) create a instantiated usb_xmas_tree**usb-components****Filename**

usb_components.pymod

Classes[usb-disk](#)[usb-santa](#)[usb-tablet-comp](#)[usb-wacom-tablet-comp](#)**Global Commands**[create-usb-disk](#) create a non-instantiated usb-disk[create-usb-santa](#) create a non-instantiated usb-santa[create-usb-tablet-comp](#) create a non-instantiated usb-tablet-comp[create-usb-wacom-tablet-comp](#) create a non-instantiated usb-wacom-tablet-comp[new-usb-disk](#) create an instantiated usb-disk[new-usb-disk-from-image](#) create new usb disk with content[new-usb-santa](#) create an instantiated usb-santa[new-usb-tablet-comp](#) create an instantiated usb-tablet-comp[new-usb-wacom-tablet-comp](#) create an instantiated usb-wacom-tablet-comp

usb-disk

Filename

usb-disk.so

Classes

[usb_disk](#)

usb-hid-components

Filename

usb_hid_components.pymod

Classes

[usb_hs_keyboard_comp](#)
[usb_keyboard_comp](#)
[usb_mouse_comp](#)

Global Commands

[create-usb-hs-keyboard-comp](#)
[create-usb-keyboard-comp](#)
[create-usb-mouse-comp](#)
[new-usb-hs-keyboard-comp](#)
[new-usb-keyboard-comp](#)
[new-usb-mouse-comp](#)

create a a non-instantiated usb_hs_keyboard_comp
create a a non-instantiated usb_keyboard_comp
create a a non-instantiated usb_mouse_comp
create a an instantiated usb_hs_keyboard_comp
create a an instantiated usb_keyboard_comp
create a an instantiated usb_mouse_comp

usb-input-devices

Filename

usb-input-devices.so

Classes

[usb_hs_keyboard](#)
[usb_keyboard](#)
[usb_mouse](#)

usb-santa

Filename

usb-santa.so

Classes

[usb_santa](#)

usb-tablet

Filename

usb-tablet.so

Classes

[usb_tablet](#)

usb-wacom-tablet

Filename

usb-wacom-tablet.so

Classes

[usb_wacom_tablet](#)

vga-pci

Filename

vga-pci.so

Classes

[vga_pci](#)

Haps

[Vga_Break_String](#)

vmcom

Filename

vmcom.so

Classes

[vmcom](#)

wdb-remote

Filename

wdb-remote.pymod

Classes

[wdb-remote](#)

Global Commands

[new-wdb-remote](#) create a wdb session

x11-console**Filename**

x11-console.so

Classes

[gfx-console](#)

Haps

[Gfx_Break_String](#)

[Graphics_Console_New_Title](#)

[Graphics_Console_Show_Hide](#)

xterm-console**Filename**

xterm-console.so

Classes

[text-console](#)

Haps

[Text_Console_New_Title](#)

[Text_Console_Show_Hide](#)

[Xterm_Break_String](#)

Index

Symbols

!, 113
*, 115
<, 117
<<, 117
<=, 117
>, 118
>>, 118
>=, 118
|, 311
+, 115
+=, 115
-, 115
->, 116
-=, 116
/, 116
:, 116
=, 117
==, 118
@, 118
#, 113
\$, 114
%, 114
&, 114
^, 119
[, 119
~, 311
|hyperpage, 119

A

a, 199
abi
 namespace command
 symtable, 863
abs_pointer, 928
abs_pointer_activate, 929
accel-vga, 539, 1092

accept-inquiries
 namespace command
 generic-flash-memory, 656
active-node
 namespace command
 os_awareness, 381
add
 namespace command
 state-assertion, 852
add-connector
 namespace command
 std-service-node, 505
add-data-to-script-pipe, 120
add-diff-file
 namespace command
 fc-disk, 630
 ide-disk, 719
 image, 725
 scsi-disk, 813
 simple-scsi-disk, 845
add-diff-partial-file
 namespace command
 fc-disk, 630
 ide-disk, 720
 scsi-disk, 814
 simple-scsi-disk, 846
add-directory, 120
add-host
 namespace command
 service-node, 827
 std-service-node, 505
add-map
 namespace command
 memory-space, 736
 port-space, 778
add-mem-lis
 namespace command

state-assertion, 853
 add-module-directory, 120
 add-partial-diff-file
 namespace command
 image, 725
 add-pathmap-entry, 121
 add-profiler
 namespace command
 g-cache, 648
 add-sun-partition
 namespace command
 fc-disk, 630
 ide-disk, 720
 scsi-disk, 814
 simple-scsi-disk, 846
 add-symbol-file, 121
 add-vlan
 namespace command
 ethernet_vlan_switch, 358
 addr_space_t, 1006
 address-profile-data
 namespace command
 address_profiler, 931
 address-profile-info
 namespace command
 address_profiler, 931
 address-profile-toplist
 namespace command
 address_profiler, 932
 address_profiler, 930
 agent_symdebug_interface, 1092
 alias, 121
 AM79C973, 542, 1092
 and, 122
 api-apropos, 123
 api-help, 122
 api-search, 123
 apropos, 199
 arp
 namespace command
 service-node, 827
 std-service-node, 506
 AT24Cxx, 545, 1092
 attr-meter, 547, 1093
 attribute_monitor, 933

auto-partition-configuration, 123
 auto-release
 namespace command
 gfx-console, 671

B

b, 124
 base-trace-mem-hier, 548
 BCM5703C, 550, 1093
 BCM5704C, 553, 1093
 bin, 123
 bitmask-translator, 556, 1093
 bookmark, 274
 branch_arc, 934
 branch_arc_iter_t, 934
 branch_arc_t, 934
 branch_arc_type_t, 934
 branch_recorder, 557
 branch_recorder_direction_t, 934
 break, 124
 namespace command
 breakpoint, 937
 gfx-console, 671
 telnet_console, 877
 text-console, 887
 break-cr, 125
 namespace command
 int_register, 1001
 break-enter
 namespace command
 os_awareness, 382
 break-exception, 126
 break-exit
 namespace command
 os_awareness, 382
 break-hap, 126
 break-io, 127
 break-line, 127
 namespace command
 tcf-context-proxy, 874
 break-location, 128
 namespace command
 tcf-context-proxy, 875
 break-log, 128
 breakpoint, 936
 breakpoint-manager, 1093

breakpoints, 558
 bridge, 939
 bt, 283

C

c, 269

cancel_notify
 software interface method, 1052

capture-start
 namespace command
 ms1553-link, 747
 telnet_console, 877
 text-console, 887

capture-stop
 namespace command
 ms1553-link, 747
 telnet_console, 877
 text-console, 887

cb, 173
 namespace command
 cycle, 951

cba, 174
 namespace command
 cycle, 951

cc, 270

cd, 128

cell, 559

cell-and-clocks, 313

cellstat
 namespace command
 mtprof, 752

central, 1094

central-client, 561

central-server, 563

Central_Blocked, 1060

Central_Unblocked, 1060

change-namespace, 129

change-reference-clock
 namespace command
 std-etg, 470
 std-service-node, 506

check-cell-partitioning, 130

checkpoint, 940

CL-PD6729, 564, 1094

clear
 namespace command

 data-profiler, 581

clear-directories, 130

clear-memorystream, 130

clear-pathmap, 130

clear-recorder, 131

cli, 566

CLI_Command_Added, 1060

CLI_Variable_Write, 1061

clipboard-gateway, 567, 1094

clock, 568, 1094

close
 namespace command
 gfx-console, 672
 text-console, 887

close-tap-interface, 131

cn, 129

code-coverage
 namespace command
 os_awareness, 383

command-history, 131

command-list, 131

component, 316, 941

Component_Change, 1061

component_connector, 944

Component_Hierarchy_Change, 1061

configuration-shortest-paths, 132

connect, 132
 namespace command
 BCM5703C, 551
 BCM5704C, 554
 central-client, 561
 component, 317
 DEC21041, 585
 DEC21140A, 588
 DEC21143, 593
 eth-transceiver, 619
 i2c2543, 705
 i2c2546, 708
 mii-transceiver, 745
 service-node, 827

connect-central, 132

connect-components, 133

connect-panel-to-frontend, 133

connect-real-network, 133

connect-real-network-bridge, 134

namespace command
 ethernet-link, 624
 ethernet_cable, 340
 ethernet_hub, 346
 ethernet_switch, 352

connect-real-network-host, 135
 namespace command
 ethernet-link, 625
 ethernet_cable, 340
 ethernet_hub, 346
 ethernet_switch, 352

connect-real-network-napt, 135
 namespace command
 ethernet-link, 625
 ethernet_cable, 341
 ethernet_hub, 347
 ethernet_switch, 353
 ethernet_vlan_switch, 358

connect-real-network-port-in, 136

connect-real-network-port-out, 136

connect-real-network-router, 137
 namespace command
 ethernet-link, 626
 ethernet_cable, 341
 ethernet_hub, 347
 ethernet_switch, 353

connect-to-link
 namespace command
 std-service-node, 506

connections
 namespace command
 central-server, 563

connector, 569, 946

context, 571

continue, 269

continue-cycles, 270

continue-seconds, 271

copy-connector, 137

copyright, 138

Core_Address_Not_Mapped, 1062

Core_Asynchronous_Trap, 1062

Core_At_Exit, 1062

Core_Back_To_Front, 1063

Core_Breakpoint_Change, 1063

Core_Breakpoint_Memop, 1063

Core_Conf_Class_Register, 1064

Core_Conf_Class_Unregister, 1064

Core_Conf_Clock_Change_Cell, 1064

Core_Conf_Object_Change_Clock, 1064

Core_Conf_Object_Create, 1065

Core_Conf_Object_Created, 1065

Core_Conf_Object_Delete, 1065

Core_Conf_Object_Pre_Delete, 1066

Core_Conf_Object_Rename, 1066

Core_Conf_Objects_Created, 1066

Core_Conf_Objects_Deleted, 1066

Core_Configuration_Loaded, 1067

Core_Context_Activate, 1067

Core_Context_Change, 1067

Core_Context_Deactivate, 1068

Core_Context_Updated, 1068

Core_Continuation, 1068

Core_Control_Register_Read, 1069

Core_Control_Register_Write, 1069

Core_Cycle_Count, 1070

Core_Device_Access_Memop, 1070

Core_Disable_Breakpoints, 1070

Core_Discard_Future, 1070

Core_DSTC_Flush_Counter, 1071

Core_Exception, 1071

Core_Exception_Return, 1072

Core_External_Interrupt, 1072

Core_Frequency_Changed, 1073

Core_Global_Message, 1073

Core_Hap_Callback_Installed, 1073

Core_Hap_Callback_Removed, 1073

Core_Hap_Type_Added, 1074

Core_Image_Activity, 1074

Core_Initial_Configuration, 1074

Core_Log_Message, 1075

Core_Log_Message_Extended, 1075

Core_Magic_Instruction, 1075

Core_Memory_Space_Map_Changed, 1076

Core_Mode_Change, 1076

Core_Module_Loaded, 1076

Core_Multithreading_Changed, 1077

Core_NotImplemented, 1077

Core_Periodic_Event, 1077

Core_Preferences_Changed, 1077

Core_Processor_Schedule_Changed, 1078

- Core_Recent_Files_Changed, [1078](#)
- Core_Rexec_Active, [1078](#)
- Core_Simulation_Mode_Change, [1079](#)
- Core_Simulation_Stopped, [1079](#)
- Core_SkipTo_Progress, [1079](#)
- Core_Source_Step, [1080](#)
- Core_Step_Count, [1080](#)
- Core_Sync_Instruction, [1080](#)
- Core_Time_Transition, [1081](#)
- Core_Timing_Model_Change, [1081](#)
- Core_Workspace_Changed, [1081](#)
- Core_Write_Configuration, [1082](#)
- coverage_profiler, [578](#)
- cp3_quad100tx, [320, 1095](#)
- cpu-group, [580, 1095](#)
- cpu-switch-time, [138](#)
 - namespace command
 - cell, [559](#)
- cpumode-software-tracker, [1095](#)
- create-and-connect-ddr-memory, [138](#)
- create-cell-and-clocks, [139](#)
- create-cp3-quad100tx, [139](#)
- create-datagram-link, [140](#)
- create-ddr-memory-module, [140](#)
- create-ddr2-memory-module, [141](#)
- create-ddr3-memory-module, [141](#)
- create-dummy-component, [141](#)
- create-etg-comp, [142](#)
- create-etg-panel, [142](#)
- create-eth-injector-comp, [143](#)
- create-ethernet-cable, [143](#)
- create-ethernet-hub, [143](#)
- create-ethernet-switch, [144](#)
- create-ethernet-vlan-switch, [144](#)
- create-example-keypad, [145](#)
- create-example-status-panel, [145](#)
- create-generic-pcie-switch, [145](#)
- create-i2c-link-v2, [146](#)
- create-instruction-data-splitter, [146](#)
- create-isa-fourport, [147](#)
- create-isa-lance, [147](#)
- create-isa-vga, [147](#)
- create-memory-timer, [148](#)
- create-micron-mtfc2ggqdi-emmc-card, [148](#)
- create-micron-mtfc4ggqdi-emmc-card, [148](#)
- create-micron-mtfc4ggqdi-sdhc-card, [149](#)
- create-os-awareness, [149](#)
- create-partition
 - namespace command
 - ide-disk, [720](#)
 - scsi-disk, [814](#)
 - simple-scsi-disk, [846](#)
- create-pc-dual-serial-ports, [149](#)
- create-pc-floppy-controller, [150](#)
- create-pc-quad-serial-ports, [150](#)
- create-pc-single-parallel-port, [150](#)
- create-pci-accel-vga, [151](#)
- create-pci-am79c973, [151](#)
- create-pci-bcm5703c, [151](#)
- create-pci-bcm5704c, [151](#)
- create-pci-dec21041, [152](#)
- create-pci-dec21140a, [152](#)
- create-pci-dec21140a-dml, [152](#)
- create-pci-dec21143, [152](#)
- create-pci-i21152, [153](#)
- create-pci-i82543gc, [153](#)
- create-pci-i82546bg, [153](#)
- create-pci-i82559, [154](#)
- create-pci-ispl040, [154](#)
- create-pci-ispl2200, [154](#)
- create-pci-pd6729, [154](#)
- create-pci-pmc1553, [155](#)
- create-pci-sil680a, [155](#)
- create-pci-sym53c810, [155](#)
- create-pci-sym53c875, [155](#)
- create-pci-sym53c876, [156](#)
- create-pci-tsb12lv26, [156](#)
- create-pci-vga, [156](#)
- create-phy-comp, [157](#)
- create-phy-mii-transceiver, [157](#)
- create-ps2-keyboard-mouse, [157](#)
- create-rapidio-link, [158](#)
- create-real-network-bridge, [158](#)
- create-real-network-host, [158](#)
- create-real-network-router, [159](#)
- create-sample-gcache, [159](#)
- create-script-barrier, [159](#)
- create-script-pipe, [160](#)
- create-sdram-memory-module, [160](#)
- create-ser-link, [160](#)

create-signal-link, 161
 create-simple-fc-disk, 161
 create-simple-memory-module, 161
 create-sio-w83627hf, 162
 create-std-etg, 162
 create-std-ethernet-link, 162
 create-std-firewire-bus, 163
 create-std-firewire-sample-device, 163
 create-std-generic-link, 163
 create-std-generic-link-sample, 164
 create-std-graphics-console, 164
 create-std-host-serial-console, 164
 create-std-ide-cdrom, 164
 create-std-ide-disk, 165
 create-std-mmc-card, 165
 create-std-ms1553-link, 165
 create-std-pcmcia-flash-disk, 166
 create-std-sata-cdrom, 166
 create-std-sata-disk, 166
 create-std-scsi-bus, 167
 create-std-scsi-cdrom, 167
 create-std-scsi-disk, 167
 create-std-serial-link, 168
 create-std-server-console, 168
 create-std-service-node, 168
 create-std-super-io, 168
 create-std-telnet-console, 169
 create-std-text-console, 169
 create-std-text-graphics-console, 169
 create-sun-vtoc-header
 namespace command
 fc-disk, 630
 ide-disk, 721
 scsi-disk, 815
 simple-scsi-disk, 847
 create-sun-vtoc-partition
 namespace command
 fc-disk, 631
 ide-disk, 721
 scsi-disk, 815
 simple-scsi-disk, 847
 create-unconnected-ethernet-probe, 170
 create-usb-disk, 170
 create-usb-hs-keyboard-comp, 170
 create-usb-keyboard-comp, 171
 create-usb-mouse-comp, 171
 create-usb-santa, 171
 create-usb-tablet-comp, 172
 create-usb-wacom-tablet-comp, 172
 create-usb-xmas-tree, 172
 current-namespace, 172
 current-processor, 173
 cycle, 948
 cycle-break, 173
 namespace command
 cycle, 951
 cycle-break-absolute, 173
 namespace command
 cycle, 951

D

da, 181
 data-profiler, 581
 datagram-link, 1095
 datagram_link, 322, 953
 datagram_link_endpoint, 582
 datagram_link_impl, 583
 date, 174
 dbuffer_t, 957, 961, 973, 974, 993, 996
 ddr-memory-module, 324
 ddr2-memory-module, 327
 ddr3-memory-module, 330
 debug
 namespace command
 memory_space, 1016
 debug-context, 174
 debug-program
 namespace command
 component, 317
 dec, 174
 DEC21041, 584, 1096
 DEC21140A, 587, 1096
 DEC21140A-dml, 590, 1096
 DEC21143, 592, 1096
 default-port-forward-target, 175
 default-translation
 namespace command
 ide-disk, 721
 defined, 175
 del-map
 namespace command

memory-space, 736
 port-space, 778
delete, 175
 namespace command
 component, 317
 eth-probe, 610
 file-cdrom, 633
 gfx-console, 672
delete-bookmark, 176
delete-host
 namespace command
 service-node, 827
 std-service-node, 506
delete-pattern
 namespace command
 hypersim-pattern-matcher, 684
delete-sun-vtoc-partition
 namespace command
 fc-disk, 631
 ide-disk, 722
 scsi-disk, 815
 simple-scsi-disk, 848
devs, 176
dhcp-add-pool
 namespace command
 service-node, 828
 std-service-node, 506
dhcp-leases
 namespace command
 service-node, 828
 std-service-node, 507
digit-grouping, 176
dirs, 177
disable, 177
 namespace command
 mtprof, 752
 realtime, 794
 trace-sync, 897
disable-hypersim, 178
disable-input
 namespace command
 gfx-console, 672
disable-magic-breakpoint, 178
disable-mtprof, 178
disable-multithreading, 179
disable-page-sharing, 179
disable-quiet
 namespace command
 text-console, 888
disable-read-only
 namespace command
 text-console, 888
disable-real-dns
 namespace command
 service-node, 828
 std-service-node, 507
disable-real-time-mode, 180
disable-reverse-execution, 180
disable-service
 namespace command
 service-node, 828
 std-service-node, 507
disable-tracker
 namespace command
 os_awareness, 384
disable-visual-feedback
 namespace command
 gfx-console, 672
disassemble, 181
disassemble-settings, 181
disassemble_v9, 595, 1096
disassemble_x86, 596, 1097
disconnect, 182
 namespace command
 BCM5703C, 551
 BCM5704C, 554
 central-client, 561
 component, 318
 DEC21041, 585
 DEC21140A, 588
 DEC21143, 593
 eth-transceiver, 619
 gdb-remote, 653
 i2c, 705
 i2c, 708
 mii-transceiver, 746
 telnet_console, 878
disconnect-real-network, 182
 namespace command
 ethernet-link, 626

ethernet_cable, 342
 ethernet_hub, 348
 ethernet_switch, 354
 real-network-bridge, 451
 real-network-host, 453
 real-network-router, 455
 disconnect-real-network-port-in, 182
 disconnect-real-network-port-out, 183
 display, 184
 dm9161, 597, 1097
 down, 184
 dstc-disable, 184
 dstc-enable, 185
 dummy-component, 333
 dump-sun-partition
 namespace command
 fc-disk, 631
 ide-disk, 722
 scsi-disk, 816
 simple-scsi-disk, 848
 dynamic_link_connector, 599

E

echo, 186
 eject
 namespace command
 ide-cdrom, 716
 scsi-cdrom, 810
 else, 186
 empty-device-c, 601, 1097
 empty-device-cc, 602, 1097
 enable, 186
 namespace command
 mtprof, 753
 realtime, 795
 trace-sync, 897
 enable-core2-bugfix, 187
 enable-ftp-alg
 namespace command
 service-node, 828
 std-service-node, 507
 enable-hypersim, 187
 enable-input
 namespace command
 gfx-console, 673
 enable-magic-breakpoint, 187
 enable-mtprof, 188
 enable-multithreading, 188
 enable-page-sharing, 188
 enable-quiet
 namespace command
 text-console, 888
 enable-read-only
 namespace command
 text-console, 888
 enable-real-dns
 namespace command
 service-node, 829
 std-service-node, 507
 enable-real-time-mode, 189
 enable-reverse-execution, 190
 enable-service
 namespace command
 service-node, 829
 std-service-node, 508
 enable-tracker
 namespace command
 os_awareness, 384
 enable-visual-feedback
 namespace command
 gfx-console, 673
 env, 190
 etg, 603, 1097
 etg_comp, 335
 etg_panel, 337
 eth-cable-link, 605
 eth-cable-link-endpoint, 606
 eth-hub-link, 607
 eth-hub-link-endpoint, 608
 eth-injector, 1098
 eth-link-snoop-endpoint, 609
 eth-links, 1098
 eth-probe, 610, 1099
 eth-probe-port, 613
 eth-switch-link, 614
 eth-switch-link-endpoint, 615
 eth-switch-link-snoop-endpoint, 617
 eth-transceiver, 619, 1099
 eth_injector, 621
 eth_injector_comp, 338, 1099
 ethereal, 309

namespace command
 eth-probe, 612
 ethernet_cable, 344
 ethernet_hub, 350
 ethernet_switch, 356
 ethernet_vlan_switch, 360
ethereal-stop, 309
 namespace command
 eth-probe, 612
 ethernet_cable, 344
 ethernet_hub, 350
 ethernet_switch, 356
 ethernet_vlan_switch, 360
ethernet-link, 622, 1100
ethernet_cable, 339, 954
ethernet_common, 955
ethernet_device, 957
Ethernet_Frame, 1082
ethernet_hub, 345
ethernet_link, 960
ethernet_probe, 963
ethernet_snoop, 965
ethernet_switch, 351
ethernet_vlan_snoop, 966
ethernet_vlan_switch, 357
example-keypad, 362
example-status-panel, 364
except, 190
exec, 191
exec-info, 191
exec-info-clean, 191
exit, 263
expect, 191
extended_serial, 967

F

f, 193
fake-space, 628
fc-disk, 629, 1100
FC_SCSI_Command, 1082
fforward
 namespace command
 state-assertion, 853
file-cdrom, 633, 1100
file-exists, 192
fin, 192

namespace command
 context, 572
find
 namespace command
 os_awareness, 384
finish, 192
 namespace command
 context, 572
finish-connection
 namespace command
 real-network-bridge, 451
 real-network-host, 453
 rn-eth-bridge-tap, 803
finish-function, 192
 namespace command
 context, 571
firewire-bus, 1100
firewire-components, 1101
firewire-sample-device, 1101
firewire_bus, 634, 968
firewire_device, 970
firewire_device_test_wrapper, 636
Firewire_Reset, 1083
firewire_sample_device, 637
Firewire_Transfer, 1083
flash-memory, 638, 1101
follow-context
 namespace command
 gdb-remote, 653
foreach, 192
frags_t, 994, 997
frame, 193
frequency-bus, 1101
frequency_bus, 639
frequency_listener, 972
frontend-server, 641, 1102
frontend-server-console, 642, 1102
ftp-alg, 643
ftp-control, 644
ftp-data, 645
ftp-service, 646, 1102
ftp_alg, 1102
function-profile, 193

G

g-cache, 648, 1103

g-link, 1103
 gdb-remote, 651, 1103
 generic-flash-memory, 654
 generic-message-link, 658
 generic-message-sample-device, 660, 1103
 generic mmc-card, 661, 1104
 generic-pcie-switch-comp, 1104
 generic-pcie-switch-port, 1104
 generic-spi-flash, 1104
 generic_eth_phy, 662, 1104
 generic_message_device, 973
 Generic_Message_Frame, 1083
 generic_message_link, 974
 generic_pcie_switch, 366
 generic_pcie_switch_port, 664
 generic_spi_flash, 666
 get, 193

- namespace command
- memory-space, 737
- port-space, 778

 get-breakpoint-list, 194
 get-class-list, 194
 get-component-list, 195
 get-component-object

- namespace command
- component, 318

 get-component-prefix, 195
 get-connection

- namespace command
- component, 318

 get-connector-list

- namespace command
- component, 318

 get-error-command, 196
 get-error-file, 196
 get-error-line, 196
 get-error-message, 197
 get-free-connector

- namespace command
- ethernet_cable, 342
- ethernet_hub, 348
- ethernet_switch, 354
- ethernet_vlan_switch, 358

 get-free-trunk-connector

- namespace command

 ethernet_vlan_switch, 358
 get-object-list, 197
 get-processor-list

- namespace command
- component, 318

 get_current_nodes

- software interface method, 1051

 get_node

- software interface method, 1051

 get_subtree

- software interface method, 1051

 gfx-console, 668
 Gfx_Break_String, 1084
 grab-setup

- namespace command
- gfx-console, 673

 Graphics_Console_New_Title, 1084
 Graphics_Console_Show_Hide, 1084
 gui, 676

H

h, 198
 hap-meter, 677, 1105
 hard-reset

- namespace command
- PMC1553, 773

 help, 198
 help-search, 199
 hex, 200
 hl, 212
 host-condor-pci-1553, 679, 1105
 host-pmc-1553, 680, 1105
 host-serial-console, 681, 1105
 host-source-at

- namespace command
- symtable, 863

 hostfs, 683, 1106
 hypersim-pattern-matcher, 684, 1106
 hypersim-status, 200

I

i21150, 686, 1106
 i21152, 688, 1106
 i21154, 690, 1106
 i21554, 1107
 i21554-prim, 692

i21554-scnd, 693
 i21555, 1107
 i21555-prim, 694
 i21555-scnd, 695
 i2c-bus, 696, 1107
 i2c-link-endpoint, 698
 i2c-link-impl, 699
 i2c-link-v1, 1107
 i2c-link-v2, 1108
 i2c_bridge, 977
 i2c_bus, 978
 i2c_device, 980
 i2c_link, 981
 i2c_link_v1, 700
 i2c_link_v2, 367
 i2c_master, 986
 i2c_master_v2, 987
 i2c_slave, 988
 i2c_slave_v2, 990
 i2c_slave_v2_to_bus_adapter, 702
 i2c_wire, 703
 i82543, 704, 1108
 i82546, 707, 1108
 i82559, 710, 1108
 ib, 208
 id-splitter, 712, 1108
 ide, 713, 1109
 ide-cdrom, 715
 ide-disk, 718
 ieee_802_3_mac, 993
 ieee_802_3_mac_v3, 994
 ieee_802_3_phy, 995
 ieee_802_3_phy_v2, 996
 ieee_802_3_phy_v3, 997
 if, 200
 ifm, 202
 ignore, 201
 image, 724, 998
 in, 201
 in-list, 201
 info
 namespace command
 accel-vga, 539
 AM79C973, 543
 AT24Cxx, 546
 attr-meter, 547
 BCM5703C, 551
 BCM5704C, 555
 bitmask-translator, 556
 cell, 559
 cell-and-clocks, 314
 central-client, 562
 central-server, 563
 CL-PD6729, 564
 clipboard-gateway, 567
 clock, 568
 component, 319
 connector, 570
 context, 572
 coverage_profiler, 578
 cp3_quad100tx, 320
 cpu-group, 580
 datagram_link, 322
 datagram_link_endpoint, 582
 datagram_link_impl, 583
 ddr-memory-module, 325
 ddr2-memory-module, 328
 ddr3-memory-module, 331
 DEC21041, 585
 DEC21140A, 588
 DEC21143, 593
 dm9161, 598
 dummy-component, 334
 dynamic_link_connector, 600
 empty-device-c, 601
 empty-device-cc, 602
 etg, 603
 etg_comp, 335
 etg_panel, 337
 eth-cable-link, 605
 eth-cable-link-endpoint, 606
 eth-hub-link, 607
 eth-hub-link-endpoint, 608
 eth-link-snoop-endpoint, 609
 eth-probe, 610
 eth-switch-link, 614
 eth-switch-link-endpoint, 615
 eth-switch-link-snoop-endpoint, 617
 eth-transceiver, 620
 eth_injector, 621

eth_injector_comp, 338
 ethernet-link, 627
 ethernet_cable, 342
 ethernet_hub, 348
 ethernet_switch, 354
 ethernet_vlan_switch, 359
 example-keypad, 363
 example-status-panel, 365
 fc-disk, 631
 firewire_bus, 634
 firewire_sample_device, 637
 frequency_bus, 640
 ftp-control, 644
 ftp-data, 645
 ftp-service, 647
 g-cache, 649
 gdb-remote, 653
 generic-flash-memory, 656
 generic mmc-card, 661
 generic_eth_phy, 663
 generic_PCIE_switch, 366
 generic_PCIE_switch_port, 665
 generic_spi_flash, 667
 hap-meter, 677
 hostfs, 683
 hypersim-pattern-matcher, 684
 i21150, 686
 i21152, 688
 i21154, 691
 i2c-bus, 697
 i2c-link-endpoint, 698
 i2c-link-impl, 699
 i2c_link_v1, 701
 i2c_link_v2, 367
 i2c_slave_v2_to_bus_adapter, 702
 i2c_wire, 703
 i82543, 706
 i82546, 709
 i82559, 711
 ide, 714
 ide-cdrom, 716
 ide-disk, 722
 image, 725
 instruction-data-splitter, 369
 isa-fourport, 371
 isa-lance, 373
 isa-vga, 375
 ISP1040, 728
 ISP2200, 730
 mem-traffic-meter, 733
 memory-space, 737
 memory-timer, 377
 micron_mtfc2ggqdi_emmc_card, 378
 micron_mtfc4ggqdi_emmc_card, 379
 micron_mtfc4ggqdi_sdhc_card, 380
 mii-management-bus, 743
 mii-transceiver, 746
 ms1553-link, 748
 mtprof, 753
 NS16450, 755
 NS16550, 757
 NS16550_c, 759
 onfi_flash, 763
 os_awareness, 385
 pc-dual-serial-ports, 393
 pc-floppy-controller, 395
 pc-quad-serial-ports, 397
 pc-single-parallel-port, 399
 pci-accel-vga, 401
 pci-am79c973, 403
 pci-bcm5703c, 405
 pci-bcm5704c, 407
 pci-bus, 766
 pci-dec21041, 409
 pci-dec21140a, 411
 pci-dec21140a-dml, 413
 pci-dec21143, 415
 pci-i21152, 417
 pci-i82543gc, 419
 pci-i82546bg, 421
 pci-i82559, 423
 pci-ispl040, 425
 pci-ispl2200, 427
 pci-pd6729, 429
 pci-pmc1553, 431
 pci-sil680a, 433
 pci-sym53c810, 435
 pci-sym53c875, 437
 pci-sym53c876, 439
 pci-tsb12lv26, 441

pci-vga, 443
 pcie-bus, 768
 persistent-ram, 770
 phy-mii-transceiver, 445
 phy_comp, 446
 PMC1553, 773
 port-space, 779
 preferences, 785
 ps2-keyboard-mouse, 448
 ram, 786
 rapidio_link, 449
 rapidio_link_endpoint, 787
 rapidio_link_impl, 789
 rapidio_tape, 791
 real-network-bridge, 451
 real-network-host, 453
 real-network-router, 455
 realtime, 795
 recorder, 796
 remote_sync_domain, 798
 remote_sync_node, 799
 remote_sync_server, 800
 rn-eth-bridge-raw, 802
 rn-eth-bridge-tap, 803
 rn-eth-proxy-raw, 805
 rn-ip-router-raw, 806
 rom, 807
 sample-gcache, 457
 scsi-bus, 808
 scsi-cdrom, 810
 scsi-disk, 816
 sdram-memory-module, 459
 selfprof, 818
 ser-link-endpoint, 819
 ser-link-impl, 820
 ser_link, 461
 serial-link, 822
 service-node, 829
 service-node-device, 832
 set-memory, 833
 signal-bus, 834
 signal_link, 462
 signal_link_endpoint, 835
 signal_link_impl, 837
 signal_to_interrupt, 838
 SIL680A, 839
 sim, 844
 simple-fc-disk, 465
 simple-scsi-disk, 848
 simple_memory_module, 466
 sio-w83627hf, 468
 slaver, 850
 software_tracker, 851
 state-assertion, 853
 static_link_connector, 856
 std-etg, 470
 std-ethernet-link, 472
 std-firewire-bus, 474
 std-firewire-sample-device, 476
 std-generic-link, 478
 std-generic-link-sample, 480
 std-graphics-console, 482
 std-host-serial-console, 485
 std-ide-cdrom, 487
 std-ide-disk, 489
 std-ms1553-link, 491
 std-pcmcia-flash-disk, 493
 std-scsi-bus, 495
 std-scsi-cdrom, 497
 std-scsi-disk, 499
 std-serial-link, 501
 std-server-console, 503
 std-service-node, 508
 std-super-io, 511
 std-telnet-console, 513
 std-text-console, 515
 std-text-graphics-console, 518
 std_mmc_card, 520
 std_sata_cdrom, 521
 std_sata_disk, 522
 SYM53C810, 859
 SYM53C875, 861
 sync_domain, 867
 system_panel_bool_in, 868
 system_panel_bool_out, 869
 system_panel_number_in, 870
 system_panel_number_out, 871
 system_panel_state_manager, 872
 tcf-agent, 873
 tcf-context-proxy, 875

telnet_console, 878
 telnet_frontend, 882
 text-console, 889
 text_panel_frontend, 894
 top-component, 525
 trace-sync, 898
 TSB12LV26, 903
 tsi500, 904
 usb-disk, 527
 usb-santa, 529
 usb-tablet-comp, 531
 usb-wacom-tablet-comp, 533
 usb_hs_keyboard, 908
 usb_hs_keyboard_comp, 534
 usb_keyboard, 911
 usb_keyboard_comp, 535
 usb_mouse, 914
 usb_mouse_comp, 536
 usb_tablet, 917
 usb_wacom_tablet, 919
 usb_xmas_tree, 537
 vga_pci, 921
 vmcom, 923
 wdb-remote, 926
 info-breakpoints, 208
input
 namespace command
 gfx-console, 673
 telnet_console, 878
 text-console, 889
input-file
 namespace command
 telnet_console, 878
 text-console, 889
insert
 namespace command
 ide-cdrom, 716
 scsi-cdrom, 810
 insert-ethernet-probe, 201
 instantiate-components, 202
 instruction-data-splitter, 368
 instruction-fetch-mode, 202
 int-to-double-float, 202
 int-to-extended-double-float, 203
 int-to-quad-float, 203
 int-to-single-float, 203
 int_register, 1000
 Internal_Bookmark_List_Changed, 1084
 Internal_Micro_Checkpoint_Loaded, 1085
 Internal_SB_Barrier, 1085
 Internal_SB_Command_File, 1085
 Internal_SB_Cycle_Count, 1086
 Internal_SB_Log, 1086
 Internal_SB_Pipe, 1086
 Internal_SB_Step_Count, 1087
 Internal_SB_Time, 1087
 Internal_SB_Variable, 1087
 Internal_Time_Direction_Changed, 1088
 Internal_Time_Quantum_Changed, 1088
 interrupt-script, 203
 interrupt-script-branch, 204
 io-stats, 204
 io_memory, 1006
 iostc-disable, 204
 iostc-enable, 205
 ireg_info_t, 1000
 isa-components, 1109
 isa-fourport, 370
 isa-lance, 372
 isa-vga, 374
 ISP1040, 728, 1110
 ISP2200, 730, 1110
 istc-disable, 206
 istc-enable, 206

K

kbd-abort
 namespace command
 telnet_console, 878
 text-console, 889
key-down
 namespace command
 usb_hs_keyboard, 908
 usb_keyboard, 911
key-press
 namespace command
 usb_hs_keyboard, 908
 usb_keyboard, 911
key-up
 namespace command
 usb_hs_keyboard, 908

usb_keyboard, 911
 keyboard, 1008

L

l2p, 222
 license, 207
 links
 namespace command
 central-client, 562
 linux-autodetect-settings
 namespace command
 os_awareness, 385
 list, 207
 namespace command
 os_awareness, 386
 symtable, 863
 list-attributes, 208
 list-bookmarks, 208
 list-break-strings
 namespace command
 telnet_console, 879
 text-console, 890
 list-breakpoints, 208
 list-checkpoints, 209
 list-classes, 209
 list-components, 210
 list-debug-contexts, 210
 list-directories, 211
 list-failed-modules, 211
 list-hap-callbacks, 212
 list-haps, 212
 list-host-info
 namespace command
 service-node, 829
 std-service-node, 508
 list-hypersim-patterns, 212
 list-length, 212
 list-modules, 213
 list-namespaces, 213
 list-objects, 214
 list-objects-with-interface, 214
 list-port-forwarding-setup, 215
 list-preferences, 215
 list-script-branches, 215
 list-sections, 216
 list-segments, 216

list-variables, 216
 list-vars, 216
 listen-for-exceptions
 namespace command
 hap-meter, 677
 listen-for-hap
 namespace command
 hap-meter, 678
 load-binary, 217
 namespace command
 memory-space, 737
 load-file, 218
 namespace command
 memory-space, 738
 load-module, 218
 load-parameters
 namespace command
 os_awareness, 386
 load-persistent-state, 219
 load-symbols
 namespace command
 symtable, 864
 local, 219
 log, 219
 log-level, 220
 log-setup, 220
 log-size, 221
 log-syscalls
 namespace command
 os_awareness, 386
 log-type, 221
 log_object, 1009, 1010
 logical-to-physical, 222
 lookup-file, 265
 ls, 222

M

mac-splitter, 1110
 mac_splitter, 732
 magic-breakpoint-enabled, 223
 man, 198
 map
 namespace command
 memory-space, 739
 port-space, 779
 map_demap, 1011

map_func_t, 1006
 match-string, 223
 max, 223
 mdio45_bus, 1013
 mdio45_phy, 1014
 mem-traffic-meter, 733, 1110
 memory-comp, 1110
 memory-components, 1111
 memory-space, 734
 memory-timer, 376
 memory_space, 1015
 memory_space_interface_t, 1015
 micron_mtfc2ggqdi_emmc_card, 378
 micron_mtfc4ggqdi_emmc_card, 379
 micron_mtfc4ggqdi_sdhc_card, 380
 microwire, 1017
 microwire-eeprom, 742, 1111
 mii, 1018
 mii-management-bus, 743, 1111
 mii-transceiver, 744, 1112
 mii_management, 1019
 mil-std-1553-components, 1112
 min, 224
 mmc, 1020
 mmc-card-components, 1112
 modelstat
 namespace command
 mtprof, 753
 module-list, 213
 module-list-failed, 211
 module-list-refresh, 224
 move-object, 224
 ms1553-link, 747, 1112
 ms1553-recorder-bm, 749
 ms1553-rt, 1113
 ms1553-test-rt, 750, 1113
 ms1553_bridge_terminal, 1021
 ms1553_link, 1022
 ms1553_rt, 751
 ms1553_terminal, 1024
 mtprof, 752, 1113

N

n, 254
 namespace command
 context, 572

nand_flash, 1025
 native-path, 224
 network-helper, 225
 new-attr-meter, 225
 new-cell-and-clocks, 225
 new-central-server, 226
 new-context, 226
 new-cp3-quad100tx, 227
 new-datagram-link, 227
 new-dummy-component, 228
 new-etg, 228
 new-etg-comp, 228
 new-etg-panel, 229
 new-eth-injector-comp, 229
 new-ethernet-cable, 230
 new-ethernet-hub, 230
 new-ethernet-switch, 231
 new-ethernet-vlan-switch, 231
 new-example-keypad, 232
 new-example-status-panel, 232
 new-file-cdrom, 232
 new-flash-memory, 1113
 new-gdb-remote, 233
 new-generic-pcie-switch, 233
 new-glink, 234
 new-hap-meter, 234
 new-i2c-link-v2, 234
 new-mem-traffic-meter, 235
 new-micron-mtfc2ggqdi-emmc-card, 235
 new-micron-mtfc4ggqdi-emmc-card, 236
 new-micron-mtfc4ggqdi-sdhc-card, 236
 new-os-awareness, 236
 new-phy-comp, 237
 new-rapidio-link, 237
 new-rapidio-tape, 238
 new-real-network-bridge, 238
 new-real-network-host, 238
 new-real-network-router, 239
 new-realtime, 239
 new-remote-frontend, 239
 new-sample-gcache, 240
 new-ser-link, 240
 new-signal-link, 241
 new-simple-memory-module, 241
 new-std-etg, 241

new-std-ethernet-link, 242
 new-std-firewire-bus, 242
 new-std-firewire-sample-device, 242
 new-std-generic-link, 243
 new-std-graphics-console, 243
 new-std-host-serial-console, 243
 new-std mmc-card, 244
 new-std-ms1553-link, 244
 new-std-pcmcia-flash-disk, 245
 new-std-sata-cdrom, 245
 new-std-sata-disk, 245
 new-std-scsi-bus, 246
 new-std-serial-link, 246
 new-std-server-console, 246
 new-std-service-node, 247
 new-std-telnet-console, 247
 new-std-text-console, 247
 new-std-text-graphics-console, 248
 new-symtable, 248
 new-tcf-agent, 249
 new-time-server, 249
 new-tracer, 249
 new-usb-disk, 249
 new-usb-disk-from-image, 250
 new-usb-hs-keyboard-comp, 250
 new-usb-keyboard-comp, 250
 new-usb-mouse-comp, 251
 new-usb-santa, 251
 new-usb-tablet-comp, 251
 new-usb-wacom-tablet-comp, 252
 new-usb-xmas-tree, 252
 new-wdb-remote, 252
 next, 254
 namespace command
 context, 572
 next-instruction, 254
 namespace command
 context, 572
 next-line, 254
 namespace command
 context, 572
 nexti, 254
 namespace command
 context, 572
 ni, 254

namespace command
 context, 572
 node path pattern, 382, 383, 389–391
 node-info
 namespace command
 os_awareness, 386
 node-tree
 namespace command
 os_awareness, 386
 not, 254
 notify_after_callbacks_done
 software interface method, 1052
 notify_callbacks_done
 software interface method, 1052
 notify_cpu_move_from
 software interface method, 1052
 notify_cpu_move_to
 software interface method, 1052
 notify_create
 software interface method, 1051
 notify_destroy
 software interface method, 1051
 notify_property_change
 software interface method, 1051
 notify_syscall
 software interface method, 1052
 NS16450, 755
 NS16550, 757
 NS16550_c, 759, 1114
 NS16x50, 1114

O

object-exists, 255
 oct, 255
 off
 namespace command
 context, 573
 on
 namespace command
 context, 573
 onfi-flash, 1114
 onfi_flash, 760
 open
 namespace command
 gfx-console, 674
 text-console, 890

operation_func_t, 1006
 or, 255
 os-awareness, 1114
 os_awareness, 381
 ose-detect-settings
 namespace command
 os_awareness, 387
 output-file-start, 256
 output-file-stop, 256
 output-radix, 256

P

p, 260
 packet-rate
 namespace command
 etg, 603
 packet-size
 namespace command
 etg, 604
 partition-settings
 namespace command
 os_awareness, 387
 path_translate, 1026
 pc-dual-serial-ports, 392
 pc-floppy-controller, 394
 pc-quad-serial-ports, 396
 pc-single-parallel-port, 398
 pcap-dump, 257
 namespace command
 eth-probe, 611
 ethernet_cable, 342
 ethernet_hub, 348
 ethernet_switch, 354
 ethernet_vlan_switch, 359
 pcap-dump-stop, 257
 namespace command
 eth-probe, 611
 ethernet_cable, 343
 ethernet_hub, 349
 ethernet_switch, 355
 ethernet_vlan_switch, 359
 pcapdump, 257
 pcapdump-stop, 257
 PCF8582C, 764, 1114
 pci-accel-vga, 400
 pci-am79c973, 402

pci-bcm5703c, 404
 pci-bcm5704c, 406
 pci-bus, 765, 1115
 pci-components, 1115
 pci-dec21041, 408
 pci-dec21140a, 410
 pci-dec21140a-dml, 412
 pci-dec21143, 414
 pci-header
 namespace command
 accel-vga, 540
 AM79C973, 544
 BCM5703C, 552
 BCM5704C, 555
 CL-PD6729, 564
 DEC21041, 586
 DEC21140A, 589
 DEC21140A-dml, 591
 DEC21143, 594
 generic_pcie_switch_port, 665
 i21150, 686
 i21152, 688
 i21154, 691
 i82543, 706
 i82546, 709
 i82559, 711
 ISP1040, 728
 ISP2200, 730
 PMC1553, 773
 SIL680A, 839
 SYM53C810, 859
 SYM53C875, 861
 TSB12LV26, 903
 vga_pci, 921
 pci-i21152, 416
 pci-i82543gc, 418
 pci-i82546bg, 420
 pci-i82559, 422
 pci-isp1040, 424
 pci-isp2200, 426
 pci-pd6729, 428
 pci-pmc1553, 430
 pci-sil680a, 432
 pci-sym53c810, 434
 pci-sym53c875, 436

pci-sym53c876, 438
 pci-tsb12lv26, 440
 pci-vga, 442
 pci_express, 1027
 pci_express_hotplug, 1029
 pcie-bus, 767, 1116
 pdisable, 257
 penable, 258
 peq, 260
 perfanalyze-client, 769
 persistent-ram, 770
 phy-comp, 1116
 phy-components, 1116
 phy-mii-transceiver, 444
 phy_comp, 446
 pid, 258
 pipe, 258
 plain-symbols
 namespace command
 symtable, 864
 playback-start
 namespace command
 ms1553-link, 748
 recorder, 796
 telnet_console, 879
 text-console, 890
 playback-stop
 namespace command
 ms1553-link, 748
 rapidio_tape, 791
 recorder, 796
 telnet_console, 879
 text-console, 890
 PMC1553, 771, 1117
 popd, 259
 port-forward-incoming-server, 774
 port-forward-outgoing-server, 775
 port-space, 777
 port_space, 1030
 pos, 259
 namespace command
 symtable, 864
 pow, 259
 preferences, 782
 preg, 260
 print, 260
 print-event-queue, 260
 print-partition-info
 namespace command
 ide-disk, 722
 scsi-disk, 816
 simple-scsi-disk, 848
 print-partition-table
 namespace command
 ide-disk, 722
 scsi-disk, 816
 simple-scsi-disk, 848
 print-pci-config-reg
 namespace command
 accel-vga, 540
 AM79C973, 544
 BCM5703C, 552
 BCM5704C, 555
 CL-PD6729, 564
 DEC21041, 586
 DEC21140A, 589
 DEC21140A-dml, 591
 DEC21143, 594
 generic_pcie_switch_port, 665
 i21150, 686
 i21152, 688
 i21154, 691
 i82543, 706
 i82546, 709
 i82559, 711
 ISP1040, 729
 ISP2200, 730
 PMC1553, 773
 SIL680A, 839
 SYM53C810, 860
 SYM53C875, 862
 TSB12LV26, 903
 vga_pci, 921
 print-sun-vtoc
 namespace command
 fc-disk, 632
 ide-disk, 723
 scsi-disk, 817
 simple-scsi-disk, 849
 print-time, 261

namespace command
 cycle, 952
ps2-keyboard-mouse, 447
psel, 261
pselect, 261
pstatus, 261
psym, 262
ptime, 261
 namespace command
 cycle, 952
pushd, 262
pwd, 262
python, 263

Q

q, 263
qnx-detect-settings
 namespace command
 os_awareness, 388
quiet
 namespace command
 text-console, 891
quit, 263

R

r, 269
ram, 786
range, 264
rapidio-link, 1117
rapidio-simple-device, 1117
rapidio-tape, 1117
rapidio_link, 449
rapidio_link_endpoint, 787
rapidio_link_impl, 789
rapidio_simple_device, 790
rapidio_tape, 791
rapidio_v3, 1031
rapidio_v5, 1033
rc, 270
read
 namespace command
 memory-space, 739
 port-space, 779
read-configuration, 264
read-only
 namespace command

text-console, 891
read-reg, 264
 namespace command
 int_register, 1002
read-variable, 265
readme, 265
real-network, 1118
real-network-bridge, 450
real-network-host, 452
real-network-router, 454
realtime, 793, 1118
record-start
 namespace command
 telnet_console, 879
 text-console, 891
record-stop
 namespace command
 telnet_console, 879
 text-console, 891
recorder, 796, 1036, 1119
recorder-save
 namespace command
 recorder, 797
recorder-start
 namespace command
 recorder, 797
recorder-stop
 namespace command
 rapidio_tape, 791
 recorder, 797
recorder_v2, 1037
redraw
 namespace command
 accel-vga, 540
 vga_pci, 921
refresh
 namespace command
 gfx-console, 674
refresh-rate
 namespace command
 accel-vga, 540
 vga_pci, 921
register-number
 namespace command
 int_register, 1002

release
 software interface method, 1053

remote_callback, 1038

remote_sync_domain, 798

remote_sync_node, 799

remote_sync_server, 800

remove-profiler
 namespace command
 g-cache, 649

request
 software interface method, 1053

reset-cache-lines
 namespace command
 g-cache, 649

reset-statistics
 namespace command
 g-cache, 649

resolve-file, 265

restart-simics, 266

rev, 266

rev-execution, 801

reverse, 266

reverse-cycles, 266

reverse-next-instruction, 267
 namespace command
 context, 573

reverse-next-line, 267
 namespace command
 context, 573

reverse-step-instruction, 267
 namespace command
 context, 574

reverse-step-line, 268
 namespace command
 context, 574

reverse-to, 268

reverse-until-activated
 namespace command
 context, 574

reverse-until-active
 namespace command
 context, 575

reverse-until-deactivated
 namespace command
 context, 575

revto, 268

rexec-limit, 269

Rexec_Limit_Exceeded, 1088

rlimit, 269

rn, 267
 namespace command
 context, 573

rn-eth-bridge-raw, 802

rn-eth-bridge-tap, 803

rn-eth-proxy-raw, 805

rn-ip-router-raw, 806

rnext, 267
 namespace command
 context, 573

rnexti, 267
 namespace command
 context, 573

rni, 267
 namespace command
 context, 573

rom, 807

root
 namespace command
 hostfs, 683

root_node_id
 software interface method, 1051

route
 namespace command
 service-node, 829
 std-service-node, 508

route-add
 namespace command
 service-node, 829
 std-service-node, 508

rs, 268
 namespace command
 context, 574

rs232_device, 1039

rsi
 namespace command
 context, 574

rstep, 268
 namespace command
 context, 574

rstepi, 267

namespace command
 context, 574
run, 269
run-command-file, 269
run-cycles, 270
run-python-file, 271
run-seconds, 271
run-until-activated
 namespace command
 context, 575
run-until-active
 namespace command
 context, 575
run-until-deactivated
 namespace command
 context, 576

S

s, 288
 namespace command
 context, 576
sample-gcache, 456
save
 namespace command
 coverage_profiler, 578
 image, 726
save-bmp
 namespace command
 gfx-console, 674
save-break-xy
 namespace command
 gfx-console, 674
save-component-template, 271
save-data
 namespace command
 mtprof, 754
save-diff-file
 namespace command
 fc-disk, 632
 ide-disk, 723
 image, 726
 scsi-disk, 817
 simple-scsi-disk, 849
save-persistent-state, 272
save-png
 namespace command

gfx-console, 674
 save-preferences, 272
 sb, 286
 sba, 286
 sc, 287
 scale_factor_listener, 1040
 script-branch, 273
 script-pipe-has-data, 273
 scsi-bus, 808, 1119
 scsi-cdrom, 809, 1119
 scsi-disk, 812, 1119
 SCSI_CDROM_Command, 1088
 SCSI_Disk_Command, 1089
 sdram-memory-module, 458
 search, 199
 select-profiles, 273
 selfprof, 818, 1120
 ser-link, 1120
 ser-link-endpoint, 819
 ser-link-impl, 820
 ser_link, 461
 serial-link, 821, 1120
 serial_device, 1041
 serial_link, 1042
 serial_peripheral_interface_slave, 1043
 service-node, 823, 1120
 service-node-device, 831
set, 273
 namespace command
 image, 726
 memory-space, 739
 port-space, 780
set-bookmark, 274
set-component-prefix, 275
set-context, 275
set-goal-latency
 namespace command
 ethernet_cable, 343
 ethernet_hub, 349
 ethernet_switch, 355
 ethernet_vlan_switch, 359
set-memory, 833, 1121
set-memory-limit, 275
set-min-latency, 276
set-pattern, 276

set-pc, 276
 set-prefix, 277
 set-substr, 277
 set-tftp-directory
 namespace command
 service-node, 830
 std-service-node, 508
 set-thread-limit, 278
 set_property
 software interface method, 1052
 shell, 278
 show-memorystatus, 278
 show-pathmap, 279
 si, 287
 namespace command
 context, 576
 signal, 1044
 namespace command
 gdb-remote, 653
 signal-bus, 834, 1121
 signal-link, 1121
 signal-to-interrupt, 1122
 signal_link, 462
 signal_link_endpoint, 835
 signal_link_impl, 837
 signal_to_interrupt, 838
 signed, 279
 signed16, 279
 signed32, 280
 signed64, 280
 signed8, 280
 SIL680A, 839
 sil680a, 1122
 sim, 841
 sim-break, 286
 sim-break-absolute, 286
 SIM_INTERFACE, 927
 Simics Core, 1122
 simple-fc-disk, 464
 simple-scsi-disk, 845, 1132
 simple_dispatcher, 1046
 simple_interrupt, 1047
 simple_memory_module, 466
 simulation-replaying, 281
 simulation-reversing, 281
 simulation-running, 281
 sio-w83627hf, 467
 skip-to, 282
 slaver, 850
 slaver_message, 1048
 snoop_memory, 1049
 soft-reset
 namespace command
 PMC1553, 773
 software, 1050
 software-tracker, 1132
 software-tracker-iface, 1132
 software_interface_t, 1050
 software_tracker, 851
 source-path
 namespace command
 symtable, 865
 split-string, 282
 stack-trace, 283
 start
 namespace command
 base-trace-mem-hier, 548
 etg, 604
 eth_injector, 621
 state-assertion, 853
 state-assertion, 852, 1133
 state-assertion-connect, 283
 state-assertion-create-file, 284
 state-assertion-open-file, 284
 state-assertion-receive, 284
 state-assertion-simple-assert, 285
 state-assertion-simple-record, 285
 static_link_connector, 855
 statistics
 namespace command
 g-cache, 649
 status
 namespace command
 accel-vga, 540
 AM79C973, 544
 AT24Cxx, 546
 attr-meter, 547
 BCM5703C, 552
 BCM5704C, 555
 bitmask-translator, 556

cell-and-clocks, 314
 CL-PD6729, 564
 clipboard-gateway, 567
 clock, 568
 component, 319
 connector, 570
 context, 576
 coverage_profiler, 579
 cp3_quad100tx, 320
 datagram_link, 322
 datagram_link_endpoint, 582
 datagram_link_impl, 583
 ddr-memory-module, 326
 ddr2-memory-module, 328
 ddr3-memory-module, 331
 DEC21041, 586
 DEC21140A, 589
 DEC21143, 594
 dm9161, 598
 dummy-component, 334
 dynamic_link_connector, 600
 empty-device-c, 601
 empty-device-cc, 602
 etg, 604
 etg_comp, 335
 etg_panel, 337
 eth-cable-link, 605
 eth-cable-link-endpoint, 606
 eth-hub-link, 607
 eth-hub-link-endpoint, 608
 eth-link-snoop-endpoint, 609
 eth-switch-link, 614
 eth-switch-link-endpoint, 616
 eth-switch-link-snoop-endpoint, 617
 eth-transceiver, 620
 eth_injector, 621
 eth_injector_comp, 338
 ethernet-link, 627
 ethernet_cable, 343
 ethernet_hub, 349
 ethernet_switch, 355
 ethernet_vlan_switch, 359
 example-keypad, 363
 example-status-panel, 365
 firewire_bus, 635
 firewire_sample_device, 637
 frequency_bus, 640
 ftp-control, 644
 ftp-data, 645
 ftp-service, 647
 g-cache, 649
 generic-flash-memory, 656
 generic mmc-card, 661
 generic_eth_phy, 663
 generic_pcie_switch, 366
 generic_pcie_switch_port, 665
 generic_spi_flash, 667
 hap-meter, 678
 hostfs, 683
 hypersim-pattern-matcher, 684
 i21150, 687
 i21152, 689
 i21154, 691
 i2c-bus, 697
 i2c-link-endpoint, 698
 i2c-link-impl, 699
 i2c_link_v1, 701
 i2c_link_v2, 367
 i2c_slave_v2_to_bus_adapter, 702
 i2c_wire, 703
 i82543, 706
 i82546, 709
 i82559, 711
 ide, 714
 image, 726
 instruction-data-splitter, 369
 isa-fourport, 371
 isa-lance, 373
 isa-vga, 375
 ISP1040, 729
 ISP2200, 731
 mem-traffic-meter, 733
 memory-space, 740
 memory-timer, 377
 micron_mtfc2ggqdi_emmc_card, 378
 micron_mtfc4ggqdi_emmc_card, 379
 micron_mtfc4ggqdi_sdhc_card, 380
 mii-management-bus, 743
 mii-transceiver, 746
 ms1553-link, 748

mtprof, 754
 NS16450, 756
 NS16550, 758
 NS16550_c, 759
 onfi_flash, 763
 os_awareness, 388
 pc-dual-serial-ports, 393
 pc-floppy-controller, 395
 pc-quad-serial-ports, 397
 pc-single-parallel-port, 399
 pci-accel-vga, 401
 pci-am79c973, 403
 pci-bcm5703c, 405
 pci-bcm5704c, 407
 pci-dec21041, 409
 pci-dec21140a, 411
 pci-dec21140a-dml, 413
 pci-dec21143, 415
 pci-i21152, 417
 pci-i82543gc, 419
 pci-i82546bg, 421
 pci-i82559, 423
 pci-isp1040, 425
 pci-isp2200, 427
 pci-pd6729, 429
 pci-pmc1553, 431
 pci-sil680a, 433
 pci-sym53c810, 435
 pci-sym53c875, 437
 pci-sym53c876, 439
 pci-tsb12lv26, 441
 pci-vga, 443
 persistent-ram, 770
 phy-mii-transceiver, 445
 phy_comp, 446
 PMC1553, 773
 port-forward-outgoing-server, 775
 port-space, 780
 preferences, 785
 ps2-keyboard-mouse, 448
 ram, 786
 rapidio_link, 449
 rapidio_link_endpoint, 787
 rapidio_link_impl, 789
 rapidio_tape, 792
 real-network-bridge, 451
 real-network-host, 453
 real-network-router, 455
 realtime, 795
 recorder, 797
 remote_sync_domain, 798
 remote_sync_node, 799
 remote_sync_server, 800
 rn-eth-bridge-raw, 802
 rn-eth-bridge-tap, 803
 rn-eth-proxy-raw, 805
 rn-ip-router-raw, 806
 rom, 807
 sample-gcache, 457
 scsi-bus, 808
 scsi-cdrom, 811
 scsi-disk, 817
 sdram-memory-module, 459
 selfprof, 818
 ser-link-endpoint, 819
 ser-link-impl, 820
 ser_link, 461
 serial-link, 822
 service-node, 830
 service-node-device, 832
 set-memory, 833
 signal-bus, 834
 signal_link, 462
 signal_link_endpoint, 835
 signal_link_impl, 837
 signal_to_interrupt, 838
 SIL680A, 839
 sim, 844
 simple-fc-disk, 465
 simple_memory_module, 466
 sio-w83627hf, 468
 slaver, 850
 software_tracker, 851
 state-assertion, 854
 static_link_connector, 856
 std-etg, 470
 std-ethernet-link, 472
 std-firewire-bus, 474
 std-firewire-sample-device, 476
 std-generic-link, 478

std-generic-link-sample, 480
 std-graphics-console, 482
 std-host-serial-console, 485
 std-ide-cdrom, 487
 std-ide-disk, 489
 std-ms1553-link, 491
 std-pcmcia-flash-disk, 493
 std-scsi-bus, 495
 std-scsi-cdrom, 497
 std-scsi-disk, 499
 std-serial-link, 501
 std-server-console, 503
 std-service-node, 509
 std-super-io, 511
 std-telnet-console, 513
 std-text-console, 515
 std-text-graphics-console, 518
 std_mmc_card, 520
 std_sata_cdrom, 521
 std_sata_disk, 522
 SYM53C810, 860
 SYM53C875, 862
 sync_domain, 867
 system_panel_bool_in, 868
 system_panel_bool_out, 869
 system_panel_number_in, 870
 system_panel_number_out, 871
 system_panel_state_manager, 872
 tcf-agent, 873
 tcf-context-proxy, 875
 telnet_console, 880
 telnet_frontend, 883
 text-console, 891
 text_panel_frontend, 894
 top-component, 525
 trace-sync, 898
 TSB12LV26, 903
 tsi500, 904
 usb-disk, 527
 usb-santa, 529
 usb-tablet-comp, 531
 usb-wacom-tablet-comp, 533
 usb_hs_keyboard, 909
 usb_hs_keyboard_comp, 534
 usb_keyboard, 912
 usb_keyboard_comp, 535
 usb_mouse, 914
 usb_mouse_comp, 536
 usb_tablet, 917
 usb_wacom_tablet, 919
 usb_xmas_tree, 537
 vga_pci, 921
 vmcom, 924
 wdb-remote, 926
 status-panel, 1133
 status-panel-view, 1133
 status_panel, 857
 status_panel_view, 858
 stc-status, 285
 std-components, 1133
 std-etg, 469
 std-ethernet-link, 471
 std-firewire-bus, 473
 std-firewire-sample-device, 475
 std-generic-link, 477
 std-generic-link-sample, 479
 std-graphics-console, 481
 std-host-serial-console, 484
 std-ide-cdrom, 486
 std-ide-disk, 488
 std-ms1553-link, 490
 std-pcmcia-flash-disk, 492
 std-scsi-bus, 494
 std-scsi-cdrom, 496
 std-scsi-disk, 498
 std-serial-link, 500
 std-server-console, 502
 std-service-node, 504
 std-super-io, 510
 std-telnet-console, 512
 std-text-console, 514
 std-text-graphics-console, 517
 std_mmc_card, 520
 std_sata_cdrom, 521
 std_sata_disk, 522
 step, 288
 namespace command
 context, 576
 step-break, 286
 step-break-absolute, 286

step-cycle, 287
 step-instruction, 287
 namespace command
 context, 576
 step-line, 287
 namespace command
 context, 576
 stepi, 287
 namespace command
 context, 576
 stop, 288
 namespace command
 base-trace-mem-hier, 548
 etg, 604
 state-assertion, 854
 switch-to-graphics-console
 namespace command
 std-text-graphics-console, 518
 switch-to-host-serial-console
 namespace command
 std-telnet-console, 513
 std-text-console, 516
 telnet_console, 880
 text-console, 892
 switch-to-serial-link
 namespace command
 host-serial-console, 681
 telnet_console, 880
 text-console, 892
 switch-to-telnet-console
 namespace command
 host-serial-console, 681
 std-host-serial-console, 485
 std-text-console, 516
 text-console, 892
 switch-to-text-console
 namespace command
 gfx-console, 675
 host-serial-console, 682
 std-graphics-console, 482
 std-host-serial-console, 485
 std-telnet-console, 513
 telnet_console, 880
 sym, 291
 sym-address, 288
 sym-file, 289
 sym-function, 289
 sym-line, 289
 sym-source, 289
 sym-string, 290
 sym-type, 290
 sym-value, 290
 SYM53C810, 859, 1135
 SYM53C875, 861, 1135
 symtable, 863, 1054, 1136
 namespace command
 context, 577
 Symtable_Updated, 1089
 symval, 291
 sync-info, 291
 sync_domain, 867
 system-info, 291
 system-panel, 1136
 system-panel-interface, 1136
 system-panel-text, 1137
 system-perfmeter, 292, 1137
 system-perfmeter-plot, 293
 system_panel_bool_in, 868
 system_panel_bool_out, 869
 system_panel_number_in, 870
 system_panel_number_out, 871
 system_panel_state_manager, 872

T

target
 namespace command
 gdb-remote, 653
 tbreak
 namespace command
 breakpoint, 938
 tcf-agent, 873, 1137
 tcf-context-proxy, 874
 tcpdump, 294
 namespace command
 eth-probe, 611
 ethernet_cable, 343
 ethernet_hub, 349
 ethernet_switch, 355
 ethernet_vlan_switch, 360
 tcpdump-stop, 294
 namespace command

eth-probe, 611
 ethernet_cable, 343
 ethernet_hub, 349
 ethernet_switch, 355
 ethernet_vlan_switch, 360
tcpip-info
 namespace command
 service-node, 830
 std-service-node, 509
 telnet-console, 1137
 telnet-frontend, 295, 1138
 telnet_console, 876
 telnet_frontend, 882
 terminal_frontend, 884
 text-console, 885
 text-dump
 namespace command
 accel-vga, 541
 vga_pci, 922
 Text_Console_New_Title, 1089
 Text_Console_Show_Hide, 1090
 text_panel_frontend, 894
 time-server, 895
 time-server-c, 1138
 timing-components, 1138
 timing_model, 1056
 top-component, 524
 trace, 1139
 trace-breakpoint, 295
 trace-cr, 295
 namespace command
 int_register, 1002
 trace-exception, 296
 trace-hap, 297
 trace-io, 297
 trace-io-setup, 298
 trace-mem-hier, 896
 trace-start
 namespace command
 base-trace-mem-hier, 548
 trace-stop
 namespace command
 base-trace-mem-hier, 549
 trace-sync, 897, 1139
 track
 namespace command
 os_awareness, 388
 trans-sorter, 899
 trans-splitter, 900
 trans-staller, 901, 1139
 translate, 1057
 try, 298
 TSB12LV26, 902, 1139
 tsi500, 904, 1140
 turbo-disable-block-profile, 298
 turbo-enable-block-profile, 298
 turbo-info, 299
 turbo-info-clean, 299
 turbo-show-block-profile, 299
U
ui
 namespace command
 context, 574
 UI_Run_State_Changed, 1090
 uint64_state, 1058
 unbreak, 300
 namespace command
 os_awareness, 389
 telnet_console, 880
 text-console, 892
 unbreak-cr, 300
 namespace command
 int_register, 1003
 unbreak-exception, 301
 unbreak-hap, 301
 unbreak-id
 namespace command
 telnet_console, 881
 text-console, 892
 unbreak-io, 302
 uncall, 302
 namespace command
 context, 577
 uncall-function, 302
 namespace command
 context, 577
 undisplay, 302
 unload-module, 303
 unset, 303
 unstep-instruction, 267

namespace command
 context, 574

untrace-breakpoint, 303

untrace-cr, 304

 namespace command
 int_register, 1004

untrace-exception, 304

untrace-hap, 305

untrace-io, 305

up, 306

usb-comp, 1140

usb-components, 1140

usb-disk, 526, 1141

usb-hid-components, 1141

usb-input-devices, 1141

usb-santa, 528, 1141

usb-tablet, 1142

usb-tablet-comp, 530

usb-wacom-tablet, 1142

usb-wacom-tablet-comp, 532

usb_disk, 905

usb_hs_keyboard, 907

usb_hs_keyboard_comp, 534

usb_keyboard, 910

usb_keyboard_comp, 535

usb_mouse, 913

usb_mouse_comp, 536

usb_santa, 915

usb_tablet, 916

usb_wacom_tablet, 918

usb_xmas_tree, 537

V

version, 306

vga-pci, 1142

Vga_Break_String, 1090

vga_pci, 920

vmcom, 923, 1142

vxworks-detect-settings

 namespace command
 os_awareness, 389

W

wait-for-activated

 namespace command
 os_awareness, 390

wait-for-active

 namespace command
 os_awareness, 390

wait-for-breakpoint, 306

wait-for-clients

 namespace command
 central-server, 563

wait-for-cycle

 namespace command
 cycle, 952

wait-for-deactivated

 namespace command
 os_awareness, 390

wait-for-hap, 307

wait-for-inactive

 namespace command
 os_awareness, 391

wait-for-register-read

 namespace command
 int_register, 1004

wait-for-register-write

 namespace command
 int_register, 1004

wait-for-script-barrier, 307

wait-for-script-pipe, 307

wait-for-string

 namespace command
 accel-vga, 541
 telnet_console, 881
 text-console, 893
 vga_pci, 922

wait-for-time

 namespace command
 cycle, 952

wait-for-variable, 308

wait-then-write

 namespace command
 telnet_console, 881
 text-console, 893

wdb-remote, 925, 1142

where, 283

whereis, 308

 namespace command
 symtable, 865

while, 308

win-about, 308
wireshark, 309
 namespace command
 eth-probe, 611
 ethernet_cable, 344
 ethernet_hub, 350
 ethernet_switch, 356
 ethernet_vlan_switch, 360
wireshark-stop, 309
 namespace command
 eth-probe, 612
 ethernet_cable, 344
 ethernet_hub, 350
 ethernet_switch, 356
 ethernet_vlan_switch, 360
wr-hypervisor-detect-settings
 namespace command
 os_awareness, 391
write
 namespace command
 memory-space, 740
 port-space, 780
write-configuration, 309
write-reg, 310
 namespace command
 int_register, 1005

X

x, 310
 namespace command
 image, 726
 memory-space, 741
x11-console, 1143
x86_cpuid, 1059
xterm-console, 1143
Xterm_Break_String, 1091