Xen scripts

Introduction

This page was started to encapsulate the results of reverse-engineering the as-installed Xen 4.1 /etc/xen/scripts/* scripts on Debian 7 Wheezy.

It assumes the xl toolkit is being used.

This page is published early – before extensive internal usage – because no similar information was found on the 'net.

How the scripts are called

Normally the scripts are called by xl (and thence libxenlight?) because they are named in /etc/xen/<DomU name>.cfg.

Structure of the scripts

The scripts are written in bash.

The scripts use a lot of common components: bash scripts which are sourced mostly to define functions. For example, here is the start of the vif-bridge call tree.

```
vif-bridge
|
+-- . /etc/xen/scripts/vif-common.sh
    |
    +-- . /etc/xen/scripts/xen-hotplug-common.sh
    |   |
    |   +-- . /etc/xen/scripts/hotplugpath.sh
    |   |
    |   +-- . /etc/xen/scripts/logging.sh
    |   |
    |   +-- . /etc/xen/scripts/xen-script-common.sh
    |   |
    |   +-- . /etc/xen/scripts/locking.sh
    |
    +-- . xen-network-common.sh
    |
    +-- findCommand
    |
    +-- evalVariables
    |
```

Developing the scripts

Run-time values

From vif-common.sh comments (Xen 4.1 version. The "given by the Xend global configuration" is obsolete?):

> *Parameters may be read from the environment, the command line arguments, and the store, with overriding in that order. The environment is given by the driver, the command line is given by the Xend global configuration, and store details are given by the per-domain or per-device configuration.*

Environment and arguments display

During development it may be convenient to run scripts other than by the normal mechanism, perhaps to run at a command prompt. In that case the environment must be created and normal arguments passed. A script's environment and arguments can be found by inserting this code at the beginning of a script:

```
log_fn=/tmp/${0##*/}.$$
exec >>$log_fn 2>&1
echo "env output: $(env)"
i=0
for arg in "$@"
do
    echo "Arg $((i++)): '$arg'"
done
unset i arg
```

⚠️ That scrippet leaves stdout and stderr redirected to the log file. Usually that is convenient. In case it is not, the original redirections can be saved and restored:

```
exec 3>&1; exec 4>&2    # Save current redirections
log_fn=/tmp/${0##*/}.$$
exec >>$log_fn 2>&1
echo "env output: $(env)"
i=0
for arg in "$@"
do
    echo "Arg $((i++)): '$arg'"
done
unset i arg
exec >&3; exec 2>&4    # Restore original redirections
exec 3>&-; exec 4>&-   # Close temporary file descriptors
```

Logging

Function log is available soon after a script initialises. Normally it writes to /var/log/syslog.

```
log <level> <message>
```

<level> can be debug, info, warn, error or any of the other levels documented on the logger man page.

<message> can be multiple words.

Tracing

Normally bash tracing could be enabled by adding +x to the shebang line. This did not work on Debian 7 Wheezy with Xen 4.1.

Using `set -x` in the scripts did work but stderr is sometimes discarded so it may be necessary to set up stderr redirection to log as described in the" Environment and arguments display" section above.

Why doesn't the script run?

Maybe it did run but failed with a syntax error which was not reported. Running it at a command prompt would check the syntax.

Xen may require the script's full path in the configuration file.

Is the script executable?

Functions

Name	Functionality	Defined in
_setup_bridge_port		xen-network-common.sh
_xenstore_write	xenstore-write "$@"	xen-hotplug-common.sh
add_to_bridge	If /sys/class/net/$bridge/brif/$dev does not exist, run `brctl addif $bridge $dev`	xen-network-common.sh
canonicalise_mode		block-common.sh

create_bridge		xen-network-common.sh
device_major_minor		block-common.sh
do_or_die	Run args as command. If it returns non-0 rc, call fatal	xen-hotplug-common.sh
do_without_error	Run args as command, discarding stderr. If it returns non-0 rc, log and continue.	xen-hotplug-common.sh
dom0_ip	Print the IP address of the interface in dom0 through which we are routing. This is the IP address on the interface specified as "netdev" as a parameter to these scripts, or eth0 by default.	vif-common.sh
ebusy		block-common.sh
evalVariables	If any args contain a character from ">=1", eval it	xen-script-common.sh
fatal	Log error and exit	xen-hotplug-common.sh
findCommand	If any args do not contain "=" set command to it and return	xen-script-common.sh
find_dhcpd_arg_file		xen-network-common.sh
find_dhcpd_conf_file		xen-network-common.sh
find_dhcpd_init_file		xen-network-common.sh
first_file		xen-network-common.sh
frob_iptable		vif-common.sh
handle_iptable	If iptables working, run frob_iptable function with various args	vif-common.sh
ifdown	Only defined if there is no ifup (sic) command, when it is a dummy and always returns non-zero	xen-network-common.sh
ifup	Only defined if there is no ifup command, when it is a dummy and always returns non-zero	xen-network-common.sh
ip_of	Print the IP address currently in use at the given interface	vif-common.sh
log	Log by `logger -p daemon.<level>` ... or, if that fails, to stderr	logging.sh
preiftransfer	Dummy; always returns 0	xen-network-common.sh
same_vm		block-common.sh
setup_physical_bridge_port	_setup_bridge_port $1 0	xen-network-common.sh
setup_virtual_bridge_port	_setup_bridge_port $1 1	xen-network-common.sh
sigerr	ERR trap handler. Calls fatal	xen-hotplug-common.sh
success	Tell DevController that backend is "connected"	xen-hotplug-common.sh
vtpm_add_and_activate		vtpm-common.sh
vtpm_create		vtpm-common.sh

vtpm_create_instance		vtpm-common.sh
vtpm_delete		vtpm-common.sh
vtpm_delete_instance		vtpm-common.sh
vtpm_domid_from_name		vtpm-common.sh
vtpm_get_create_reason		vtpm-common.sh
vtpm_isLocalAddress		vtpm-common.sh
vtpm_migrate		vtpm-common.sh
vtpm_migrate_local		vtpm-common.sh
vtpm_migrate_recover		vtpm-common.sh
vtpm_migration_step		vtpm-common.sh
vtpm_recover		vtpm-common.sh
vtpm_remove_instance		vtpm-common.sh
vtpm_resume		vtpm-common.sh
vtpm_setup		vtpm-common.sh
vtpm_start		vtpm-common.sh
vtpm_suspend		vtpm-common.sh
vtpm_uuid_by_domid		vtpm-common.sh
vtpm_uuid_from_vmname		vtpm-common.sh
vtpmdb_add_instance		vtpm-common.sh
vtpmdb_find_instance		vtpm-common.sh
vtpmdb_get_free_instancenum		vtpm-common.sh
vtpmdb_is_free_instancenum		vtpm-common.sh
vtpmdb_remove_entry		vtpm-common.sh
vtpmdb_validate_entry		vtpm-common.sh
write_dev		block-common.sh
xenstore_read	xenstore-read "$@"	xen-hotplug-common.sh
xenstore_read_default	xenstore-read "$1" || echo "$2"	xen-hotplug-common.sh

xenstore_write	_xenstore_write "$@" || fatal	xen-hotplug-common.sh