Search

Spice User Manual

Introduction

Spice is an open remote computing solution, providing client access to remote displays and devices (e.g. keyboard, mouse, audio). The main use case is to get remote access to virtual machines, although other use cases are possible and in various development stage.

Spice provides a desktop-like user experience, while trying to offload most of the intensive CPU and GPU tasks to the client. The basic building blocks of Spice are:

- Spice Server
- Spice Client
- Spice Protocol

The following sections provide basic information on Spice components and features, obtaining, building installing and using Spice.

Spice and Spice-related components

Spice Server

Spice server is implemented in libspice, a VDI pluggable library. Currently, the main user of this library is QEMU. QEMU uses spice-server to provide remote access to virtual machines through the Spice protocol. Virtual Device Interface (VDI) defines a set of interfaces that provide a standard way to publish virtual devices (e.g. display device, keyboard, mouse) and enables different Spice components to interact with those devices. On one side, the server communicates with the remote client using the Spice protocol and on the other side, it interacts with the VDI host application (e.g QEMU).

Spice Client

The Spice client is a program which is used by the end user to access remote systems through Spice. The recommended client is remote-viewer (which is shipped with virt-viewer). GNOME Boxes can also be used as a Spice client. spicec is an obsolete legacy client, and spicy is only a test application.

QXL Device and Drivers

Spice server supports the QXL VDI interface. When libspice is used with QEMU, a specific video PCI device can be used for improving remote display performance and enhancing the graphic capabilities of the guest graphic system. This video device is called a QXL device and requires guest QXL drivers for full functionality. However, standard VGA is supported when no driver exists.

Spice Agent

The Spice agent is an optional component for enhancing user experience and performing guest-oriented management tasks. For example, the agent injects mouse position and state to the guest when using client mouse mode. It also enables you to move cursor freely between guest and client. Other features of agent are shared clipboard (copy and paste between guest and host) and aligning guest resolution with client when entering fullscreen mode.

VDI Port Device

The Spice protocol supports a communication channel between the client and the agent on the server side. When using QEMU, Spice agent resides on the guest. VDI port is a QEMU PCI device used for communication with the agent.

Spice Protocol

The Spice protocol defines the messages and rules for the communication between the various Spice components.

Features

Multiple Channels

The server and client communicate via channels. Each channel is dedicated to a specific type of data. The available channels are the following:

Main
control and configuration

Display
graphics commands images and video streams

Inputs
keyboard and mouse inputs

Cursor
pointer device position and cursor shape

Playback
audio received from the server to be played by the client

Record
audio captured on the client side

Smartcard
passthrough of smartcard data from the client machine to the guest OS

USB
redirection of USB devices plugged into the client to the guest OS

Image Compression

Spice offers several image compression algorithms, which can be chosen on server initiation and dynamically at run-time. Quic is a Spice proprietary image compression technology based on the SFALIC algorithm. The Lempel-Ziv (LZ) algorithm is another option. Both Quic and LZ are local algorithms encoding each image separately. Global LZ (GLZ) is another proprietary Spice technology that uses LZ with history-based global dictionary. GLZ takes advantage of repeating patterns among images to shrink the traffic and save bandwidth, which is critical in a WAN environment. Spice also offers an automatic mode for compression selection per image, where the choice between LZ/GLZ and Quic is heuristically based on image properties. Conceptually, synthetic images are better compressed with LZ/GLZ and real images are better with Quic.

Video Compression

Spice uses loss-less compression for images sent to the client. However, video streams are handled differently. Spice server heuristically identifies video areas and sends them as a video stream coded using M-JPEG. This handling saves a lot of traffic, improving Spice performance, especially in a WAN environment. However, in some circumstances the heuristic behavior might cause low quality images (e.g. identifying updated text area as a video stream). Video streaming can be chosen on server initiation and dynamically at run-time.

Mouse modes

Spice supports two mouse modes: server and client. The mode can be changed dynamically and is negotiated between the client and the server.

Server mouse
When a user clicks inside the Spice client window, the client mouse is captured and set invisible. In this mode, the server controls the mouse position on display. However, it might be problematic on WAN or on a loaded server, where mouse cursor might have some latency or non-responsiveness.

Client mouse
Not captured and is used as the effective pointing device. To enable client mouse, the VDI host application must register an absolute pointing device (e.g. USB tablet in QEMU). This mode is appropriate for WAN or for a loaded server, since cursor has smooth motion and responsiveness. However, the cursor might lose synchronization (position and shape) for a while.

Other Features

Multiple Monitors
any number of monitors is supported

Arbitrary Resolution
when using the QXL driver, the resolution of the guest OS will be automatically adjusted to the size of the client window.

USB Redirection
Spice can be used to redirect USB devices that are plugged in the client to the guest OS. This redirection can either be automatic (all newly plugged devices are redirected), or manual (the user selects which devices (s)he wants to redirect).

Smartcard Redirection
data from smartcard that are inserted into the client machine can be passed through to the guest OS. The smartcard can be used by both the client OS and the guest OS.

Bidirectional Audio
Spice supports audio playback and recording. Playback is compressed using the OPUS algorithm

Lip-sync
between video and audio. Available only when video streaming is enabled.

Migration
switching channel connectivity for supporting server migration

Pixmap and Palette caching

image data is cached on the client to avoid sending the same data

Using Spice

I'll use `qemu-kvm` as a name for the executable. If you're using a manually built qemu or a qemu without kvm then just replace `qemu-kvm` with your own binary. I'll use `host$`, `client$` and `guest$` shell prompt notations to distinguish where the command should be the command. See section [glossary] to be sure that you know difference between the host, client and guest. You can ignore the difference between guest, client and host if they are all running on the same machine.

Running qemu manually

The first thing to do is to create a guest image. You can use any raw device such as a clean logical volume, or an iSCSI lun. You may also use a file as the disk image for the guest. I'll use a file created by `qemu-img` as a demonstration.

The following command will allocate a 10GB file. See `qemu-img` man page for further information.

```
host$ qemu-img create /path/to/xp.img 10G
```

Now that we created an image, we can now start with image population. I assume that you have a locally stored ISO of your favourite operating system so you can use it for installation.

```
host$ qemu-kvm -machine vmport=off \
        -boot order=dc -vga qxl \
        -spice port=3001,disable-ticketing -soundhw hda \
        -device virtio-serial -chardev spicevmc,id=vdagent,debug=0,name=vdagent \
        -device virtserialport,chardev=vdagent,name=com.redhat.spice.0 \
        -cdrom /path/to/your.iso /path/to/your.img
```

Let's take a brief look at the qemu options that were used. The option `-machine vmport=off` disables VMWare IO port emulation, which is necessary for server mode mouse to work properly with spice. The option `-boot order=dc` specifies that the guest system should try to boot from the first cdrom and then fallback to the first disk, `-vga qxl` specifies that qemu uses a qxl graphics device.

The Spice `port` option defines what port will be used for communication with the client. The Spice option `disable-ticketing` is specifying that ticketing (simple authentication method) is not used. The virtio and chardev devices are required by the guest agent.

The `-soundhw hda` option provides an audio device for the guest to use for audio playback and recording. In order for spice audio to work properly, qemu must use the *spice* audio driver. Depending on how qemu was built, however, this might not be the default audio driver. To ensure qemu uses the spice audio driver, you can set the environment variable `QEMU_AUDIO_DRV=spice`.

Basic configuration

This section will assume that you already have a running QEMU virtual machine, and that you are running it either through virt-manager, libvirt or through direct QEMU use, and that you want to enable Spice support for this virtual machine.

Using virt-manager

Double-click on the virtual machine you are interested in, go to "View/Details". If the left pane has a "Display Spice" entry, then the virtual machine already has Spice support, and you can check the connection details (port number) by clicking on it. If it has no Spice entry, click on "Add Hardware", and add a "Graphics" element of type "Spice server". If the host and the client are not the same machine, you should check the "Listen on all public network interfaces" checkbox, otherwise you don't need to make any changes.

You should also add a QXL video device. It can be done by double-clicking on a virtual machine, then by going to View/Details, and by clicking on "Add Hardware" if the virtual machine does not have a "Video QXL" item in its left pane. From the "Add hardware" dialog, you should then create a "Video" device whose model is "QXL".

After stopping and restarting the virtual machine, it should be accessible with a Spice client.

You can remove non-Spice display entries and non-QXL video entries from the virtual machine configuration.

If you go to "Edit/Preferences/VM Details" in the main virt-manager window, you can set Spice graphics type as the default setting for new virtual machines.

Using libvirt

All libvirt examples will assume that the virtual machine to modify is `$vmname` and that virsh is using the correct libvirt connection by default.

To add Spice support to an existing virtual machine managed by libvirt, you need to edit it:

```
host$ virsh edit $vmname
```

and then add a Spice graphics element:

```
<graphics type='spice'/>
```

You should also add a QXL video device

```
<video>
    <model type='qxl'/>
</video>
```

After stopping and restarting the virtual machine `$vmname`, it should be accessible through Spice. You can check the connection parameters with:

```
host$ virsh domdisplay $vmname
```

Using QEMU

To enable Spice support to your virtual machine, you only need to append the following to your QEMU command line:

```
-spice port=3001,disable-ticketing
```

This will setup a Spice session listening on port 3001 exporting your virtual machine display.

You can also add a QXL device by appending `-vga qxl` to the command line.

Connecting to the guest

The following section will show you basic usage of the Spice client. The example connection will be related to the qemu instance started in the previous sections.

Be aware that the port used for spice communication (port 3001 in our case) should not be blocked by firewall. Host `myhost` is referring to the machine which is running our qemu instance.

```
client$ remote-viewer spice://myhost:3001
```

Ticketing

Spice does not support multiple connections to the same QEMU instance by default. So anybody who will connect to the same host and port can simply take over your session. You can solve this problem by using ticketing.

Ticketing is a simple authentication system which enables you to set simple tickets to a VM. Client has to authenticate before the connection can be established. See the Spice option `password` in the following examples.

Configuration

Using virt-manager

To set a Spice password for a virtual machine, go to this machine details in virt-manager, and then click on the "Display Spice" item in the left pane, and enter the ticket you want to use in the "Password" field.

Using libvirt

All you need to do is to append a `passwd` attribute to the Spice graphics node for your virtual machine:

```
<graphics type='spice' passwd='mysecretpassword'/>
```

Using QEMU

Adding a ticket with QEMU involves a slight modification of the `-spice` parameter used when running QEMU:

```
-spice port=3001,password=mysecretpassword
```

Client

When you start the client as usual, if ticketing was enabled on the host, remote-viewer will pop up a window asking for a password before starting the Spice session. It won’t be established if an incorrect ticket was passed to the client.

You might have figured out that passing tickets as a command-line option isn’t very safe. It’s not safe as everybody with access to the host can read it from the output of `ps(1)`. To prevent this, the ticket can be also set by using the QEMU console command `spice._set_ticket`.

Agent

Agent support allows better integration with the guest. For example, it allows copy and paste between the guest and the host OSes, dynamic resolution changes when the client window is resized/full-screened, file transfers through drag and drop, …

The agent is a daemon/service running in the guest OS so it must be installed if it was not installed by default during the guest OS installation. It also relies on a virtio-serial PCI device and a dedicated spicevmc char device to achieve communication between the guest and the host. These devices must be added to the virtual machine for the agent to work in the guest.

Configuration

Using virt-manager

The needed devices can be added from the virtual machine details. Click on "Add hardware" and then add a "Channel" device with type "Spice agent (spicevmc)". This will automatically add the needed virtio-serial device in addition to the spicevmc channel.

Using libvirt

Two distinct devices must be added:

- a virtio serial device (http://libvirt.org/formatdomain.html#elementsControllers)
- a spicevmc channel (http://libvirt.org/formatdomain.html#elementCharChannel)

```
<devices>
    <controller type='virtio-serial' index='0'/>
    <channel type='spicevmc'>
        <target type='virtio' name='com.redhat.spice.0'/>
    </channel>
</devices>
```

Using QEMU

Adding the following parameters to your QEMU command line will enable the needed devices for agent support in the guest OS:

```
-device virtio-serial \
-chardev spicevmc,id=vdagent,debug=0,name=vdagent \
-device virtserialport,chardev=vdagent,name=com.redhat.spice.0 \
```

USB redirection

With USB redirection, USB devices plugged into the client machine can be transparently redirected to the guest OS. This redirection can either be automatic (all newly plugged devices are redirected), or manual (the user selects which devices (s)he wants to redirect).

For redirection to work, the virtual machine must have an USB2 EHCI controller (this implies 3 additional UHCI controllers). It also needs to have Spice channels for USB redirection. The number of such channels correspond to the number of USB devices that it will be possible to redirect at the same time.

Configuration

Using virt-manager

Virtual machines created with virt-manager should have a USB controller by default. In the virtual machine details, select "Controller USB" in the left pane, and make sure its model is set to USB2. You can then click on "Add Hardware" and add as many "USB Redirection" items as the number of USB devices you want to be able to redirect simultaneously.

Using libvirt

You need to add the needed USB controllers to the libvirt XML (make sure there is no pre-existing USB controller in your virtual machine XML before doing this), as well as one Spice USB redirection channel per device you want to redirect simultaneously.

```
<controller type='usb' index='0' model='ich9-ehci1'/>
<controller type='usb' index='0' model='ich9-uhci1'>
    <master startport='0'/>
</controller>
<controller type='usb' index='0' model='ich9-uhci2'>
    <master startport='2'/>
</controller>
<controller type='usb' index='0' model='ich9-uhci3'>
    <master startport='4'/>
</controller>
<redirdev bus='usb' type='spicevmc'/>
<redirdev bus='usb' type='spicevmc'/>
<redirdev bus='usb' type='spicevmc'/>
<redirdev bus='usb' type='spicevmc'/>
```

Using QEMU

Similarly to libvirt, we need to add EHCI/UHCI controllers to QEMU command line, and we also need to add one Spice redirection channel per device we want to redirect simultaneously.

```
-device ich9-usb-ehci1,id=usb \
-device ich9-usb-uhci1,masterbus=usb.0,firstport=0,multifunction=on \
-device ich9-usb-uhci2,masterbus=usb.0,firstport=2 \
-device ich9-usb-uhci3,masterbus=usb.0,firstport=4 \
-chardev spicevmc,name=usbredir,id=usbredirchardev1 \
-device usb-redir,chardev=usbredirchardev1,id=usbredirdev1 \
-chardev spicevmc,name=usbredir,id=usbredirchardev2 \
-device usb-redir,chardev=usbredirchardev2,id=usbredirdev2 \
-chardev spicevmc,name=usbredir,id=usbredirchardev3 \
-device usb-redir,chardev=usbredirchardev3,id=usbredirdev3
```

Client

The client needs to have support for USB redirection. In remote-viewer, you can select which USB devices to redirect in "File/USB device" selection once the Spice connection is established. There are also various command line redirection options which are described when running remote-viewer with `--help-spice`.

To get USB redirection working on Windows clients, you need to install UsbDk (http://www.spice-space.org /download/windows/usbdk/)

Video Streaming

SPICE streaming allows sending an encoded video stream of the guest desktop to the client. The encoding can be done from the host (inside SPICE server) or from the guest, with the help of the SPICE streaming agent.

The streaming agent is a daemon/service running in the guest OS so it must be installed if it does not yet exist on the guest system. It relies on a dedicated spiceport char device to achieve communication between the guest and the host. This device must be added to the virtual machine as described below for the streaming agent to work in the guest.

Guest Video Encoding: Agent Configuration

Using virt-manager

The needed device can be added from the virtual machine details. Click on "Add hardware" and then add a "Channel" device with type "Spice port (spiceport)". The port should be named `org.spice-space.stream.0` , and the channel should also be `org.spice-space.stream.0` .

Using libvirt

Two distinct devices must be added:

- a virtio serial device (http://libvirt.org/formatdomain.html#elementsControllers) if there is not one yet in the domain definition
- a spiceport channel (http://libvirt.org/formatdomain.html#elementsCharSpiceport)

```
<devices>
    <controller type='virtio-serial' index='0'/>
    <channel type='spiceport'>
        <source channel='org.spice-space.stream.0'/>
        <target type='virtio' name='org.spice-space.stream.0'/>
    </channel>
</devices>
```

Using QEMU

Adding the following parameters to your QEMU command line will enable the needed devices for agent support in the guest OS:

```
-device virtserialport,bus=virtio-serial0.0,nr=1,chardev=charchannel1,id=channel1,na
-chardev spiceport,name=org.spice-space.stream.0,id=charchannel1
```

Host Video Encoding

For host video encoding, SPICE natively supports MJPEG encoding. For using further codecs, SPICE server must be compiled with GStreamer support.

Setting

SPICE video streaming parameter can take three values:

- `off` : no video detection is performed,

- `all` : any fast-refreshing window can be encoded into a video stream,
- `filter` : SPICE server adds additional filters to decide if video streaming should be activated (at the moment, only small window surfaces are skipped),

Using libvirt

The `streaming` tag must be added to Spice `graphics` :

```
<graphics type='spice' autoport='yes'>
    <streaming mode='off|all|filter'/>
</graphics>
```

Using QEMU

Adding the following parameters to your QEMU command line will enable SPICE server video encoding:

```
-spice ...,streaming-video=off|all|filter,...
```

CAC smartcard redirection

Spice has a dedicated channel for smartcard redirection, using libcacard, which currently supports limited CAC emulation.

You may consider redirecting your USB card reader instead. This is easier to setup but will prevent from sharing the smartcard with both the client and the remote simultaneously.

libcacard is actually emulating a simple CAC card, sharing the card and its certificates. It can successfully be used with the coolkey PKCS#11 module.

Configuration

Using virt-manager

In the hardware details, click on "Add Hardware", then select "Smartcard". Add a "passthrough" device type.

Using libvirt

Setup a "passthrough" smartcard of type "spicevmc" on a CCID controller:

```
<controller type='ccid' index='0'/>
<smartcard mode='passthrough' type='spicevmc'>
  <address type='ccid' controller='0' slot='0'/>
</smartcard>
```

Using QEMU

With the qemu command line, you must add a USB CCID device, and a "ccid-card-passthru" associated with a "spicevmc" channel with the name "smartcard":

```
-device usb-ccid -chardev spicevmc,name=smartcard,id=ccid -device ccid-card-passthru
```

Client

In order for the client certificates to be shared with the remote, you need a NSS database configured to access the smartcard. Please look for instructions on coolkey or NSS setup and make sure you certficates can be listed with certutil.

Most Spice clients disable smartcard support by default, and need `--spice-smartcard` or similar configuration.

Multiple monitor support

When using Spice, it’s possible to use multiple monitors. For that, the guest must have multiple QXL devices (for Windows guests), or a single QXL device configured to support multiple heads (for Linux guests).

Before following the instructions in this section, make sure your virtual machine already has a QXL device. If that is not the case, refer to this section. Your guest OS will also need to have the QXL driver installed or multiple monitor support will not work.

Once your virtual machine is using a QXL device, you don’t need to make any other change to get multiple heads in a Linux guest. The following paragraph will deal with adding multiple QXL devices to get multiple monitors in a Windows guest.

Configuration

Using virt-manager

To add an additional QXL device for Windows guests, simply go to your virtual machine details. Check that you already have a "Video QXL" device, if not, click on "Add Hardware", and add a "Video" device with model "QXL". This can also work with Linux guests if your are willing to configure X.Org to use Xinerama (instead of XRandR).

If you are using a new enough distribution (for example Fedora 19), and if your virtual machine already has a QXL device, you should not need to make any changes in virt-manager. If you are using an older distribution, you can’t do the required changes from virt-manager, you’ll need to edit libvirt XML as described on this blog post.

Using libvirt

To add an additional QXL device to your virtual machine managed by libvirt, you simply need to append a new video node whose model is QXL:

```
<video>
    <model type='qxl'/>
</video>
<video>
    <model type='qxl'/>
</video>
```

Using QEMU

To get a second QXL device in your virtual machine, you need to append `-device qxl` to your QEMU command line in addition to the `-vga qxl` that is already there:

```
-vga qxl -device qxl
```

Client

You can enable additional displays from the "View → Displays" menu in remote-viewer.

TLS

TLS support allows to encrypt all/some of the channels Spice uses for its communication. A separate port is used for the encrypted channels. When connecting through a TLS channel, the Spice client will verify the certificate sent by the host. It will check that this certificate matches the hostname it’s connecting, and that this certificate is signed by a known certificate authority (CA). This can be achieved by either getting the host certificate signed by an official CA, or by passing to the client the certificate of the authority which signed the host certificate. The latter allows the use of self-signed certificates.

Configuration

Using virt-manager

It’s not currently possible to define the CA certificate/host certificate to use for the TLS connection using virt-manager, see the next section for how to enable this using libvirt.

Using libvirt

The certificate must be specified in libvirtd configuration file in */etc/libvirt/qemu.conf* (or in *~/.config/libvirt /qemu.conf* if you are using a session libvirt). See the documentation in this file reproduced below:

```
# Enable use of TLS encryption on the SPICE server.
#
# It is necessary to setup CA and issue a server certificate
# before enabling this.
#
spice_tls = 1
```

```
# Use of TLS requires that x509 certificates be issued. The
# default it to keep them in /etc/pki/libvirt-spice. This directory
# must contain
#
#  ca-cert.pem - the CA master certificate
#  server-cert.pem - the server certificate signed with ca-cert.pem
#  server-key.pem  - the server private key
#
# This option allows the certificate directory to be changed.
#
spice_tls_x509_cert_dir = "/etc/pki/libvirt-spice"
```

Once the above is done, when the domain is running, you should get something like what is below if you are leaving Spice port allocation up to libvirt:

TODO proof-read the following section:

```
host$ virsh domdisplay
spice://127.0.0.1?tls-port=5901
host$
```

This means that the connection is possible both through TLS and without any encryption. You can edit the libvirt graphics node if you want to change that behaviour and only allow connections through TLS:

```
<graphics type='spice' autoport='yes' defaultMode='secure'/>
```

Using QEMU

QEMU expects the certificates to be named the same way as what libvirt expects in the previous paragraph. The directory where these certificates can be found is specified as options to the `-spice` command line parameters:

```
-spice port=5900,tls-port=5901,disable-ticketing,x509-dir=/etc/pki/libvirt-spice
```

Client

We need to change 2 things when starting the client:

- specify the tls port to use
- specify the CA certificate to use when verifying the host certificate

With remote-viewer, this is done this way:

```
client$ remote-viewer --spice-ca-file=/etc/pki/libvirt-spice/ca-cert.ca spice://myho
```

Generating self-signed certificates

The following script can be used to create the various certificates needed to use a TLS Spice connection. Make sure to substitute the hostname of your Spice host in the subject of the certificate signing request.

```
SERVER_KEY=server-key.pem

# creating a key for our ca
if [ ! -e ca-key.pem ]; then
    openssl genrsa -des3 -out ca-key.pem 1024
fi
# creating a ca
if [ ! -e ca-cert.pem ]; then
    openssl req -new -x509 -days 1095 -key ca-key.pem -out ca-cert.pem -utf8 -subj "
fi
# create server key
if [ ! -e $SERVER_KEY ]; then
    openssl genrsa -out $SERVER_KEY 1024
fi
# create a certificate signing request (csr)
if [ ! -e server-key.csr ]; then
    openssl req -new -key $SERVER_KEY -out server-key.csr -utf8 -subj "/C=IL/L=Raana
fi
# signing our server certificate with this ca
if [ ! -e server-cert.pem ]; then
    openssl x509 -req -days 1095 -in server-key.csr -CA ca-cert.pem -CAkey ca-key.pe
fi

# now create a key that doesn't require a passphrase
openssl rsa -in $SERVER_KEY -out $SERVER_KEY.insecure
mv $SERVER_KEY $SERVER_KEY.secure
mv $SERVER_KEY.insecure $SERVER_KEY

# show the results (no other effect)
openssl rsa -noout -text -in $SERVER_KEY
openssl rsa -noout -text -in ca-key.pem
openssl req -noout -text -in server-key.csr
openssl x509 -noout -text -in server-cert.pem
openssl x509 -noout -text -in ca-cert.pem
```

SASL

Spice server and client have support for SASL authentication. When using QEMU, */etc/sasl2/qemu.conf* will be used as a configuration file. For testing, you can use the `digest-md5` mechanism, and populate a test database using `saslpasswd2 -f /etc/qemu/passwd.db -c foo`. These files have to be readable by the QEMU process that will handle your VM.

To troubleshoot SASL issues, running `strace -e open` on the QEMU process can be a useful first step.

Configuration

Using virt-manager

It’s currently not possible to enable SASL from virt-manager.

Using libvirt

SASL support for SPICE has been added to libvirt mid-October 2013 so you need a libvirt version that was released after this date. To enable SASL, you need to add `spice_sasl = 1` in */etc/libvirt/qemu.conf* for the system libvirtd instance, and to *~/.config/libvirt/qemu.conf* for the session libvirtd instance.

Using QEMU

Using SASL with QEMU involves a slight modification of the `-spice` parameter used when running QEMU:

```
-spice port=3001,sasl
```

Client

When you start the client as usual, if SASL was enabled on the host, remote-viewer will pop up a window asking for a password before starting the Spice session. It won’t be established if an incorrect ticket was passed to the client.

Folder sharing

The Spice client can share a folder with the remote guest. By default folder sharing is disabled. Use the remote-viewer "File" → "Preferences" menu to enable it. The default shared directory is the XDG Public Share directory (ie *~/Public* if you use a regular system). You may specify a different folder with `--spice-shared-dir` client option.

Configuration

Using virt-manager

In the hardware details, click on "Add Hardware", then select "Channel". Add a "Spice port" device type with the "org.spice-space.webdav.0" name.

Using libvirt

In order to set up folder sharing, qemu needs to expose a `org.spice-space.webdav.0` virtio port, associated with a corresponding Spice port:

```
<devices>
    <channel type='spiceport'>
        <source channel='org.spice-space.webdav.0'/>
        <target type='virtio' name='org.spice-space.webdav.0'/>
    </channel>
</devices>
```

Using QEMU

In order to set up folder sharing, qemu needs to expose a `org.spice-space.webdav.0` virtio port, associated with a corresponding Spice port:

```
-device virtserialport,bus=virtio-serial0.0,nr=1,chardev=charchannel1,id=channel1,na
```

Guest configuration

Windows

In a Windows guest, you must then install spice-webdavd (https://www.spice-space.org/download/windows/spice-webdavd/) service.

Linux

With a Linux guest, you must install the spice-webdavd service (the sources are available at https://git.gnome.org/browse/phodav (https://git.gnome.org/browse/phodav)). The folder will show up in GNOME Files network places (or Nautilus). It can then be mounted and browsed in traditional applications thanks to `gvfs-fuse` .

GL acceleration (virgl)

OpenGL acceleration is currently local only (it has to go through a Unix socket) and it needs guest support. It’s currently limited to recent linux distributions (for example Fedora 24).

Host-side, you need qemu 2.6, libvirt 1.3.3 and spice 0.13.1, as well as a 4.4 Linux kernel and Mesa 11.1.

Client-side, you need spice-gtk 0.31.

Guest-side, you need Mesa 11.1 and a 4.4 Linux kernel.

Configuration

Using libvirt

You need to add a virtio-gpu video device to your virtual machine instead of QXL.

```
<video>
  <model type='virtio' heads='1'>
    <acceleration accel3d='yes'/>
  </model>
</video>
```

Then you need to enable OpenGL on your SPICE graphics node:

```
<graphics type='spice' autoport='no'>
  <gl enable='yes'/>
</graphics>
```

You don’t need any port/address as they won’t be usable with GL.

Using QEMU

You need to add a virtio-gpu device on QEMU command line, as well as enable GL with SPICE. port/tls-port/addr arguments won’t be used in this setup. You need to configure a Unix socket to connect to the VM display.

```
-device virtio-vga,virgl=on -spice gl=on,unix,addr=/run/user/1000/spice.sock
```

Connecting to the guest

Connecting to the guest when virgl is in use is slightly different than usual

If libvirt is being used

```
client$ virt-viewer -a $vmname
```

If a Unix socket has been set on QEMU command line

```
client$ remote-viewer spice+unix:///run/user/1000/spice.sock
```

Intel’s GVTg

Intel GVTg allows you to share your host GPU with the guest so that it can accelerate graphic and media operations.

Configuration

To configure your host and the vGPU device, follow this blog post (https://www.kraxel.org/blog/2018/04/vgpu-display-support-finally-merged-upstream/)

For spice configuration you have two main options:

1. Use locally, connect and configure spice similar to virGL spice configuration (set vfio display=on)
2. Use remotley, connect and configure spice similar to spice-streaming-agent (set vfio display=off)
    - For remote connection you may want to accelerate your video encoding using the configured Intel’s vGPU, check for Gstreamer’s Intel accelerated codec options.

QEMU Spice reference

TODO, incomplete

Command line options

They are covered in the QEMU online documentation (http://qemu.weilnetz.de/qemu-doc.html#index-g_t_002dspice-58). Basic syntax is `-spice <spice_options>` .

QXL command line options

- ram_size
- vram_size
- revision
- debug
- guestdebug
- cmdlog
- ram_size_mb
- vram_size_mb
- vram64_size_mb

- vgamem_mb
- surfaces

QEMU console Spice commands

- `set_password spice <password> [keep|disconnect]` Set the spice connection ticket (one time password). An empty password prevents any connection. keep/disconnect indicates what to do if a client is already connected when the command is issued.
- `expire_password`
- `client_migrate_info`
- `info spice` Show current spice state

Spice guest additions

While you will be able to remotely access your virtual machine through Spice without making any change to the virtual machine configuration, you can get better integration if you tweak it specially for Spice.

If your virtual machine has a QXL video device and you install the corrresponding guest driver, your guest will support higher resolutions, multiple monitors, resizing to arbitrary resolutions, …

Installing the Spice vdagent in your guest will let you copy and paste between your guest and client OSes, to drag and drop files between the 2 OSes, … In order for the agent to work, your virtual machine must have a virtio serial device (and the corresponding guest drivers) as well as a Spice spicevmc channel.

Windows guest

The recommended way of getting all the needed drivers installed is to use the all-in-one Spice guest tools installer which can be found on spice-space.org (http://spice-space.org/download/windows/spice-guest-tools/).

If you want to manually install them, the QXL driver can be downloaded from this location (http://spice-space.org/download/windows/qxl/) , agent builds can be found here (http://spice-space.org/download/windows/vdagent/). You also need the vioserial driver which is distributed with the other virtio-win drivers (https://alt.fedoraproject.org/pub/alt/virtio-win/latest/images/bin/).

Installation

Installing Spice on RHEL or Fedora

Be aware that RHEL has no builds of qemu/spice-server for i386, only x86_64 builds are available. RHEL >=6 and Fedora >=13

```
yum install qemu-kvm virt-viewer
```

The package spice-protocol will be downloaded automatically as a dependency of package kvm.

RHEVM Users

oVirt/RHEVM users could be also interested in the spice-xpi package as it allows you to execute spice-client directly from the oVirt/RHEVM UserPortal.

```
yum install spice-xpi
```

General build instructions

This section is for distributions that don’t have Spice packages in their repositories. It will show you step by step how to build the required Spice components.

Client requirements

See the https://gitlab.freedesktop.org/spice/spice-gtk/raw/master/README.md (https://gitlab.freedesktop.org/spice/spice-gtk/raw/master/README.md) [README file in spice-gtk] for the list of dependencies.

Host requirements

- KVM supported by kernel (It should work also without KVM, but it’s not being tested as most Linux distrubitions already support KVM.)

Guest requirements

Linux guest

spice-vdagent requires virtio-serial support to be enabled. This is described in the chapter [agent]. Guest should have installed qxl driver (xorg-x11-drv-qxl on Fedora and RHEL).

Windows guest

Drivers for QXL and drivers for virtio-serial require Win XP SP3.

Building

The environment variable `$BUILD_ROOT` will point to a directory with stored sources and will be used during the whole build process. The variable `$INST_ROOT` will point to a directory in which Spice will be installed.

These instructions may be outdated. Feel free to ask on the Spice mailing list if you need help building from source.

```
export BUILD_ROOT=/tmp/spice; mkdir $BUILD_ROOT
export INST_ROOT="/opt/spice"; mkdir $INST_ROOT
export PKG_CONFIG_PATH=$INST_ROOT/lib/pkgconfig:$PKG_CONFIG_PATH

cd $BUILD_ROOT
git clone https://gitlab.freedesktop.org/spice/spice.git
cd $BUILD_ROOT/spice
./configure --prefix=$INST_ROOT
make
make install

cd $BUILD_ROOT
git clone git://git.qemu.org/qemu.git
cd $BUILD_ROOT/qemu
./configure --prefix=$INST_ROOT --target-list=x86_64-softmmu --enable-spice
make
make install
```

Setting up PATH

Last steps before starting with Spice are to set proper `PATH` variable. For example RHEL is using /usr/libexec as directory for qemu-kvm binaries. The following setup should be suitable for qemu and Spice built according to the instructions in this chapter.

```
echo "export PATH=$PATH:$INST_ROOT/bin" >> ~/.bashrc
source ~/.bashrc
```

You should now be able to access the qemu-system-x86_64 Spice binary.

Debugging

Server side

If the virtual machine was started using QEMU directly, SPICE server logs will be output to your console stdout.

When using libvirt, logs are located in `/var/log/libvirt/qemu/` for the qemu system instance (`qemu:///system`), and in `~/.cache/libvirt/qemu/log` for the qemu session instance (`qemu:///session`).

It's possible to get more verbose output by setting the `G_MESSAGES_DEBUG` environment variable to `Spice` or `all` before starting QEMU as described in GLib documentation (https://developer.gnome.org/glib/stable /glib-running.html).

When using libvirt, the environment variable needs to be set from the XML domain definition as described in libvirt documentation (https://libvirt.org/drvqemu.html#qemucommand).

Client side

remote-viewer can be started with `SPICE_DEBUG=1` in the environment, or with `--spice-debug` in order to get it to output more logs on stdout. `SPICE_DEBUG` should work with any application using spice-gtk (virt-manager, gnome-boxes, …).

Guest side

QXL KMS driver

On recent Linux kernels using the QXL kms driver, booting the kernel with the `drm.debug=0xf` parameter. `journalctl -k` / `dmesg` will then contain debug logs for the QXL kms driver. This can also be changed at runtime by echo'ing this value to `/sys/module/drm/parameters/debug` .

qxl.guestdebug QEMU parameter

It's also possible to get some host-side debugging logs from the guest QXL driver. The driver reads a guestdebug parameter from the rom, which can be set when starting the VM. This can be enabled with `-global qxl-vga.guestdebug=3` , or `-global qxl.guestdebug=3` for secondary devices.

The corresponding libvirt XML is:

```
<domain type='kvm' xmlns:qemu='http://libvirt.org/schemas/domain/qemu/1.0'>
  ....
  <qemu:commandline>
    <qemu:arg value='-global'/>
    <qemu:arg value='qxl-vga.guestdebug=3'/>
  </qemu:commandline>
</domain>
```

(don't forget to add the `xmlns:qemu='http://libvirt.org/schemas/domain/qemu/1.0'` attribute to the toplevel `<domain>` node)

You can go up to 12 (or more, look for DEBUG_PRINT in the driver), you get really a lot of debug information. Interesting values are:

- 3 - will give you all the highlevel commands (DrvCopyBits, DrvBitBlt, etc.)
- 6 - will also show QXLGetBitMap
- 11 - will show caching of images (this is a driver cache, not to be confused with the cache shared between server and client).

qxl.cmdlog QEMU parameter

This will dump all the commands passing through the ringbuffer on the device side. It is driver and hence guest agnostic. This can be enabled with `-global qxl-vga.cmdlog=1`, or `-global qxl.cmdlog=1` for secondary devices. See the section above for the libvirt XML to use.

Agent

On Linux, `journalctl -t spice-vdagentd -t spice-vdagent` will display the agent log messages. spice-vdagent can also be restarted by hand with the `-d` argument in order to display more logs.

On Windows, the agent logs can be found in `C:\WINDOWS\TEMP\VDAGENT.LOG` and `C:\WINDOWS\TEMP\VDSERVICE.LOG`

Recording/replaying SPICE server traffic

Since spice-server 0.12.6, it's possible to record display traffic sent by the SPICE server in order to replay it afterwards for a client without needing to start a VM. This is achieved by setting the environment variable `SPICE_WORKER_RECORD_FILENAME` to the filename to write the traffic to before starting QEMU.

Once the recording session is done, the `spice-server-replay` tool can be used to replay the previously recorded SPICE session, for example:

```
spice-server-replay -p 5900 -c "remote-viewer spice://localhost:5900" recorded-sessi
```

Appendix A: Manual authors

The following people have contributed to this manual:

Arnon Gilboa
Christophe Fergeau
Lubos Kocman
Marc-André Lureau

Yaniv Kamay

Glossary

Host
Host is a machine running an instance of qemu-kvm.

Guest
Guest is a virtual machine hosted on the host which will be accessed with a Spice client.

Client
Client is referring to a system running the Spice client (the recommended one is virt-viewer).