Cloud Hosted Router (CHR)

Cloud Hosted Router (CHR) is a RouterOS version intended for running as a virtual machine. It supports the x86 64-bit architecture and can be used on most of the popular hypervisors such as VMWare, Hyper-V, VirtualBox, KVM, and others. CHR has full RouterOS features enabled by default but has a different licensing model than other RouterOS versions.

System Requirements

- Package version: RouterOS v6.34 or newer
- Host CPU: 64-bit with virtualization support
- RAM: 128MB or more (Max: 128GB)
- Disk: 128MB disk space for the CHR virtual hard drive (Max: 16GB)

The minimum required RAM depends on interface count and CPU count. You can get an approximate number by using the following formula:

- RouterOS v6 - RAM = 128 + [8 × (CPU_COUNT) × (INTERFACE_COUNT - 1)]
- RouterOS v7 - RAM = 256 + [8 × (CPU_COUNT) × (INTERFACE_COUNT - 1)]

Note: We recommend allocating at least 1024MiB of RAM for CHR instances.

CHR has been tested on the following platforms:

- VirtualBox 6 on Linux and OS X
- VMWare Fusion 7 and 8 on OS X
- VMWare ESXi 6.5
- Qemu 2.4.0.1 on Linux and OS X
- Hyper-V on Windows Server 2008r2, 2012 and Windows 10 *(Only Generation 1 Hyper-V virtual machine is supported at the moment)*

- Xen Server 7.1

Warning: Hypervisors that provide paravirtualization are not supported.

Usable Network and Disk interfaces on various hypervisors:

- ESX:
    - Network: vmxnet3, E1000
    - Disk: IDE, VMware paravirtual SCSI, LSI Logic SAS, LSI Logic Parallel

- Hyper-V:
    - Network: Network adapter, Legacy Network adapter
    - Disk: IDE, SCSI

- Qemu/KVM:
    - Network: Virtio, E1000, vmxnet3 (optional)
    - Disk: IDE, Sata, Virtio

- VirtualBox
    - Network: E1000, rtl8193
    - Disk: IDE, Sata, SCSI, SAS

Note: SCSI controller Hyper-V and ESX is usable just for secondary disks, system image must be used with IDE controller!

Warning: We do not recommend using the E1000 network interface if better synthetic interface options are available on a specific Hypervisor!

How to Install a virtual RouterOS system with CHR images

We provide 4 different virtual disk images to choose from. Note that they are only disk images, and you can't simply run them.

- RAW disk image (.img file)
- VMWare disk image (.vmdk file)
- Hyper-V disk image (.vhdx file)
- VirtualBox disk image (.vdi file)

Steps to install CHR

1. Download virtual disk image for your hypervisor
2. Create a guest virtual machine
3. Use the previously downloaded image file as a virtual disk drive
4. Start the guest CHR virtual machine
5. Log in to your new CHR. The default user is 'admin', without a password

Please note that running CHR systems can be cloned and copied, but the copy will be aware of the previous trial period, so you cannot extend your trial time by making a copy of your CHR. However, you are allowed to license both systems individually. To make a new trail system, you need to make a fresh installation and reconfigure RouterOS.

Installing CHR guides

- VMWare Fusion / Workstation and ESXi 6.5
- VirtualBox
- Hyper-V
- Amazon Web Services (AWS)
- Hetzner Cloud
- Linode
- Google Compute Engine
- ProxMox

CHR Licensing

The CHR has 4 license levels:

- ***free***
- ***p1*** *perpetual-1* ($45)
- ***p10*** *perpetual-10* ($95)

- ***p-unlimited*** *perpetual-unlimited* ($250)

The 60-day free trial license is available for all paid license levels. To get the free trial license, you have to have an account on MikroTik.com as all license management is done there.

Perpetual is a lifetime license (buy once, use forever). It is possible to transfer a perpetual license to another CHR instance. A running CHR instance will indicate the time when it has to access the account server to renew its license. If the CHR instance will not be able to renew the license it will behave as if the trial period has run out and will not allow an upgrade of RouterOS to a newer version.

After licensing a running trial system, you **must** manually run the */system license renew* function from the CHR to make it active. Otherwise, the system will not know you have licensed it in your account. If you do not do this before the system deadline time, the trial will end and you will have to do a complete fresh CHR installation, request a new trial, and then license it with the license you had obtained.

License	Speed limit	Price
Free	1Mbit	FREE
P1	1Gbit	$45
P10	10Gbit	$95
P-Unlimited	Unlimited	$250

Paid licenses

p1

p1 (perpetual-1) license level allows CHR to run indefinitely. It is limited to 1Gbps upload per interface. All the rest of the features provided by CHR are available without restrictions. It is possible to upgrade *p1* to *p10* or *p-unlimited (New license level can be purchased by standard price).* After the upgrade is purchased the former license will become available for later use on your account.

p10

p10 (perpetual-10) license level allows CHR to run indefinitely. It is limited to 10Gbps upload per interface. All the rest of the features provided by CHR are available without restrictions. It is possible to upgrade *p10* to *p-unlimited* After the upgrade is purchased the former license will become available for later use on your account.

p-unlimited

The *p-unlimited* (perpetual-unlimited) license level allows CHR to run indefinitely. It is the highest tier license and it has no enforced limitations.

Free licenses

There are several options to use and try CHR free of charge.

free

The *free* license level allows CHR to run indefinitely. It is limited to 1Mbps upload per interface. All the rest of the features provided by CHR are available without restrictions. To use this, all you have to do is download the disk image file from our download page and create a virtual guest.

60-day trial

In addition to the limited Free installation, you can also test the increased speed of P1/P10/PU licenses with a 60 trial.

You will have to have an account registered on MikroTik.com. Then you can request the desired license level for trial from your router that will assign your router ID to your account and enable the purchase of the license from your account. All the paid license equivalents are available for trial. A trial period is 60 days from the day of acquisition after this time passes, your license menu will start to show "Limited upgrades", which means that RouterOS can no longer be upgraded.

If you plan to purchase the selected license, you must do it within 60 days of the trial end date. If your trial ends, and there are no purchases within 2 months after it ended, the device will no longer appear in your MikroTik account. You will have to make a new CHR installation to make a purchase within the required time frame.

To request a trial license, you must run the command "**/system license renew**" from the CHR device command line. You will be asked for the username and password of your mikrotik.com account.

If you plan to use multiple virtual systems of the same kind, it may be possible that the next machine has the same system ID as the original one. This can happen on certain cloud providers, such as Linode. To avoid this, after your first boot, run the command "/system license generate-new-id" **before** you request a trial license. Note that this feature must be used only while CHR is running on a free type of RouterOS license. If you have already obtained a paid or trial license, do not use the regenerate feature since you will not be able to update your current key any more

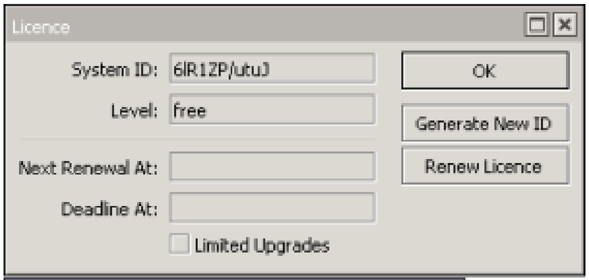
Getting the License

After the initial setup, a CHR instance will have a *free* license assigned. From there, it is possible to upgrade the license to a higher tier. Once you have a trial license all the work with the license is done on the account server where it is possible to upgrade the license to a higher tier unless it is *p-unlimited* already.

Upgrade from free to p1 or higher

Initial upgrade from the *free* tier to anything higher than that incurs CHR instance registration on the account server. To do that you have to enter your MikroTik.com username and password and the desired license level you want to acquire. As a result, a CHR ID number will be assigned to your account on the account server and a 60-day trial created for that ID. There are 2 ways to obtain a license - using WinBox or RouterOS command-line interface:

Using WinBox (Sytem -> License menu):



Licence

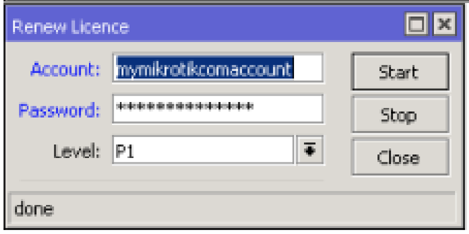
System ID: 6lR1ZP/utuJ

Level: free

Next Renewal At:

Deadline At:

□ Limited Upgrades



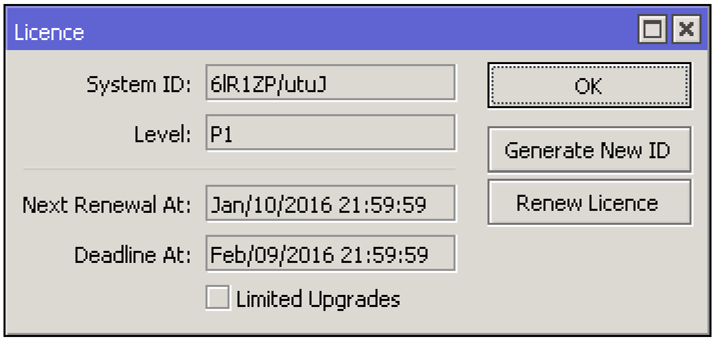
Renew Licence

Account: mymikrotikcomaccount

Password: ***************

Level: P1

done



Licence

System ID: 6lR1ZP/utuJ

Level: P1

Next Renewal At: Jan/10/2016 21:59:59

Deadline At: Feb/09/2016 21:59:59

□ Limited Upgrades

Using the command-line interface:

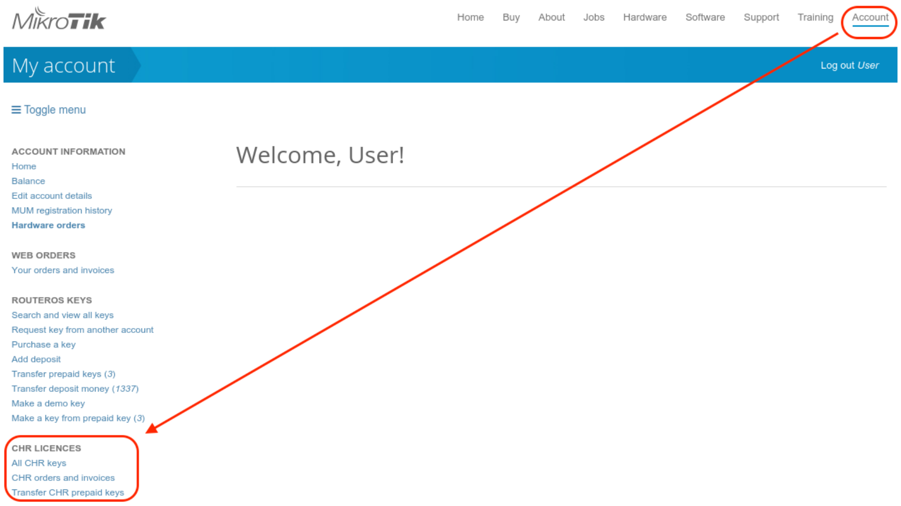
```
[admin@MikroTik] > /system license print
  system-id: 6lR1ZP/utuJ
      level: free

[admin@MikroTik] > /system license renew
account: mymikrotikcomaccount
password: *********************
level: p1
  status: done

[admin@MikroTik] > /system license print
        system-id: 6lR1ZP/utuJ
            level: p1
  next-renewal-at: jan/10/2016 21:59:59
      deadline-at: feb/09/2016 21:59:59
```

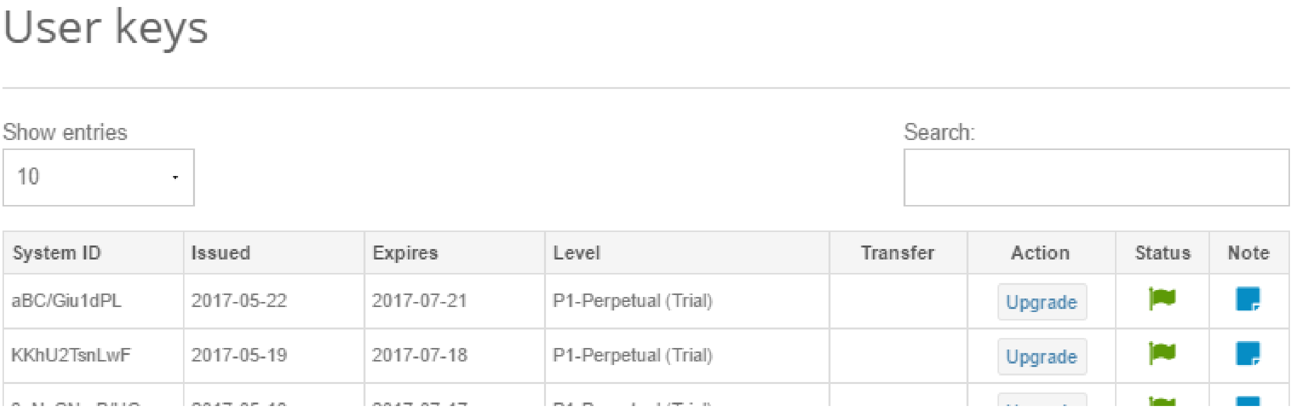
To acquire a higher level trial, set up a new CHR instance, renew the license, and select the desired level.

To upgrade from a Trial license to Paid go to MikroTik.com account server and choose 'all keys' in Cloud Hosted Router (CHR) section:



Welcome, User!

You will be presented with a list of your CHR machines and licenses:



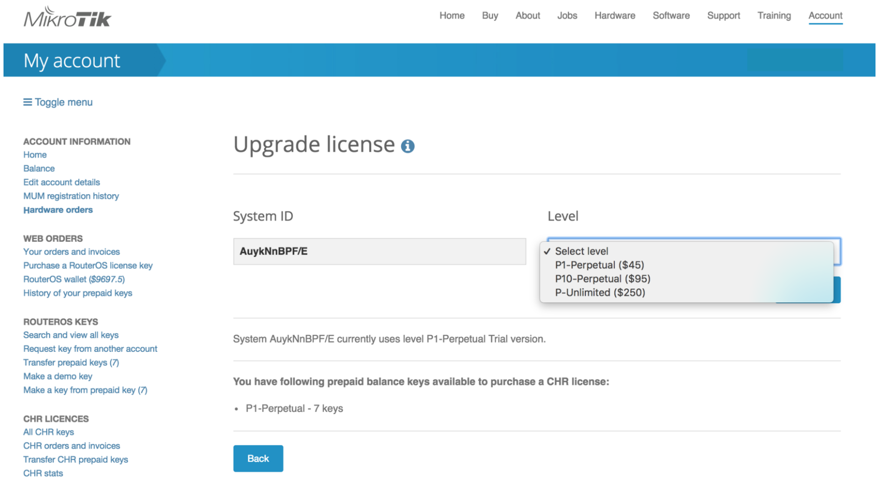
User keys

Show entries 10

Search:

System ID	Issued	Expires	Level	Transfer	Action	Status	Note
aBC/Giu1dPL	2017-05-22	2017-07-21	P1-Perpetual (Trial)		Upgrade	⚑	🟦
KKhU2TsnLwF	2017-05-19	2017-07-18	P1-Perpetual (Trial)		Upgrade	⚑	🟦
[illegible]	[illegible]	[illegible]	[illegible]		[illegible]	⚑	🟦

To upgrade from a Trial to a Paid license click 'Upgrade', choose the desired license level (it can be different than the level of the trial license), and click 'Upgrade key':



Upgrade license

System ID: AuykNnBPF/E

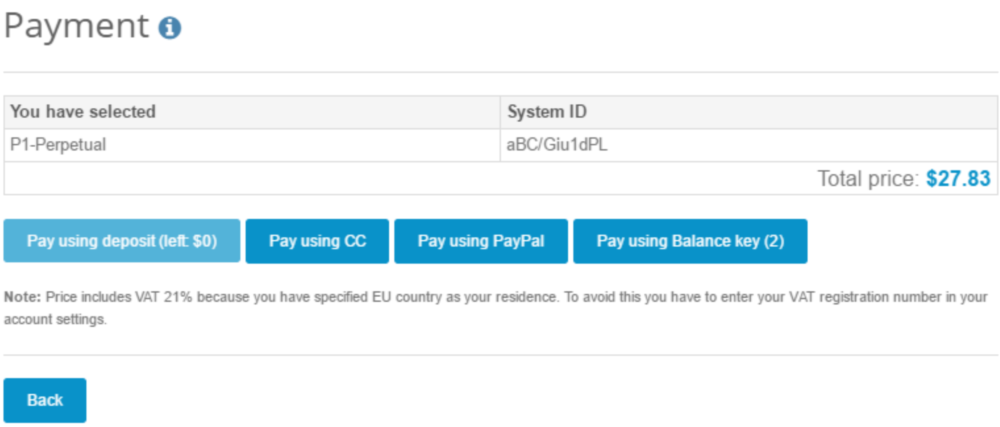
Level

System AuykNnBPF/E currently uses level P1-Perpetual Trial version.

You have following prepaid balance keys available to purchase a CHR license:

- P1-Perpetual - 7 keys

Choose the payment method:



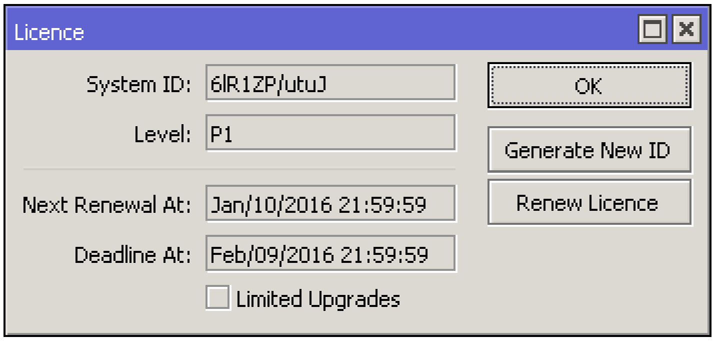
Payment

You have selected	System ID
P1-Perpetual	aBC/Giu1dPL
	Total price: **$27.83**

Note: Price includes VAT 21% because you have specified EU country as your residence. To avoid this you have to enter your VAT registration number in your account settings.

It is possible to pay using account balance (deposit), credit card (CC), PayPal, or using Balance (prepaid) key (if you have any).

License Update



System ID: 6lR1ZP/utuJ

Level: P1

Next Renewal At: Jan/10/2016 21:59:59

Deadline At: Feb/09/2016 21:59:59

☐ Limited Upgrades

In '/system license' menu router will indicate the time *next-renewal-at* when it will attempt to contact the server located on licence.mikrotik.com. Communication attempts will be performed once an hour after the date on *next-renewal-at* and will not cease until the server responds with an error. If the *deadline-at* date is reached without successfully contacting the account server, the router will consider that license has expired and will disallow further software updates. However, the router will continue to work with the same license tier as before.

Virtual Network Adapters

> ⚠ Fast Path is supported since RouterOS v7 for "vmxnet3" and "virtio-net" adapters.

RouterOS v6 does not support Fast Path.

Troubleshooting

Running on VMware ESXi

Changing MTU

VMware ESXi supports MTU of up to 9000 bytes. To get the benefit of that, you have to adjust your ESXi installation to allow a higher MTU. Virtual Ethernet interface added **after** the MTU change will be properly allowed by the ESXi server to pass jumbo frames. Interfaces added prior to MTU change on the ESXi server will be barred by the ESXi server (it will still report old MTU as maximum possible size). If you have this, you have to re-add interfaces to the virtual guests.

Example. There are 2 interfaces added to the ESXi guest, auto-detected MTU on the interfaces show MTU size as it was at the time when the interface was added:

```
[admin@chr-vm] > interface ethernet print
Flags: X - disabled, R - running, S - slave
 #    NAME            MTU MAC-ADDRESS       ARP
 0 R  ether1         9000 00:0C:29:35:37:5C enabled
 1 R  ether2         1500 00:0C:29:35:37:66 enabled
```

Using bridge on Linux

If Linux bridge supports IGMP snooping, and there are problems with IPv6 traffic it is required to disable that feature as it interacts with MLD packets (multicast) and is not passing them through.

```
echo -n 0 > /sys/class/net/vmbr0/bridge/multicast_snooping
```

Packets not passing from guests

The problem: after configuring a software interface (VLAN, EoIP, bridge, etc.) on the guest CHR it stops passing data to the outside world beyond the router.

The solution: check your VMS (Virtualization Management System) security settings, if other MAC addresses allowed to pass if packets with VLAN tags allowed to pass through. Adjust the security settings according to your needs like allowing MAC spoofing or a certain MAC address range. For VLAN interfaces, it is usually possible to define allowed VLAN tags or VLAN tag range.

Using VLANs on CHR in various Hypervisors

In some hypervisors before VLAN can be used on VMs, they need to first be configured on the hypervisor itself.

ESXI

Enable Promiscuous mode in a port group or virtual switch that you will use for specific VM.

ESX documentation:

- https://kb.vmware.com/kb/1002934
- https://kb.vmware.com/kb/1004099

Hyper-V

Hyper-V documentation:

- https://technet.microsoft.com/en-us/library/cc816585(v=ws.10).aspx#Anchor_2

bhyve hypervisor

It won't be possible to run CHR on this hypervisor. CHR cannot be run as a para-virtualized platform.

Linode

When creating multiple Linodes with the same disk size, new Linodes will have the same systemID. This will cause issues to get a Trial/Paid license. To avoid this, run the command `/system license generate-new-id` after the first boot and before you request a trial or paid license. This will make sure the ID is unique.

Some useful articles:

Specific VLAN is untagged by NIC interface:

- https://blogs.msdn.microsoft.com/adamfazio/2008/11/14/understanding-hyper-v-vlans/
- https://www.aidanfinn.com/?p=10164

Allow passing other VLANs:

- https://social.technet.microsoft.com/Forums/windows/en-US/79d36d5b-c794-4502-8ed4-b7a4183b1891/vlan-tags-and-hyperv-switches?forum=winserverhyperv

Guest tools

VMWare

Time synchronization

Must be enabled from GUI ('Synchronize guest time with host'). Backward synchronization is disabled by default - if the guest is ahead of the host by more than ~5 seconds, synchronization is not performed

Power operations

- *poweron* and *resume* scripts are executed (if present and enabled) after power on and resume operations respectively.
- *poweroff* and *suspend* scripts are executed before power off and suspend operations respectively.
- If scripts take longer than 30 seconds or contain errors, the operation fails
- In case of failure, retrying the same operation will ignore any errors and complete it successfully
- Failed script output is saved to file (e. g. 'poweroff-script.log', 'resume-script.log' etc)
- Scripts can be enabled/disabled from hypervisor GUI ('run VMware Tools Scripts') or by enbaling/disabling scripts from console

Quiescing/backup

Guest filesystem quiescing is performed only if requested.

- *freeze* script is executed before freezing the filesystem
- *freeze-fail* script is executed if hypervisor failed to prepare for a snapshot or if *freeze* script failed
- *thaw* script is executed after the snapshot has been taken
- Script run time is limited to 60 seconds
- *freeze* script timeouts and errors result in the backup operation being aborted
- FAT32 disks are not quiesced
- Failed script output is saved to file (e. g. 'freeze-script.log', 'freeze-fail-script.log', 'thaw-script.log')

Guest info

Networking, disk, and OS info are reported to hypervisor every 30 seconds (GuestStats (memory) are disabled by default, can be enabled by setting 'guestinfo.disable-perfmon = "FALSE"' in VM config).

- The order, in which network interfaces are reported, can be controlled by setting 'guestinfo.exclude-nics', 'guestinfo.primary-nics' and 'guestinfo.low-priority-nics' options. Standard wildcard patterns can be used.

Provisioning

Can use the ProcessManager from vim API to execute scripts. Python bindings are available

- Main data structure: GuestProgramSpec
  - The *workingDirectory* and *envVariables* members are ignored
  - *programPath* must be set to either 'inline' or 'import'
  - If *programPath* is **'inline'**, *arguments* are interpreted as script text
  - If *programPath* is **'import'**, *arguments* are interpreted as file path

After using *GuestProgramSpec* together with an instance of GuestAuthentication as arguments to StartProgramInGuest unique *JobID* is obtained.

Script progress can be tracked by using the ListProcessesInGuest command. *ListProcessesInGuest* accepts an array of job id's; passing an empty array will report on all jobs started from API

- *ListProcessesInGuest* returns an array of GuestProcessInfo instances:
  - *pid* field is set to *JobID*
  - *endTime* is only set after completion
  - *exitCode* is set to 0 on success and -1 on error
  - *name* is set to 'inline' or 'import' (same as *programPath* in *GuestProgramSpec*)

Information about completed jobs is kept around for ~1 minute, or until *ListProcessesInGuest* (with the corresponding *JobID*) is called. If the script fails, a file named 'vix_job_$JobID$ .txt' containing the script output is created. Script run time is limited to 120 seconds and script output is not saved on timeout,

- The vmrun command *runScriptInGuest* can also be used
- The PowerCLI cmdlet Invoke-VMScript is not supported
- Host/guest file transfer is not supported

Python example

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import sys,time
from pyVim import connect
from pyVmomi import vmodl,vim


def runInline(content,vm,creds,source):
    ''' Execute script source on vm '''
    if isinstance(source, list):
        source = '\n'.join(source)
    ps = vim.vm.guest.ProcessManager.ProgramSpec(
                programPath = 'console',
                arguments = source
        )
    return content.guestOperationsManager.processManager.StartProgramInGuest(vm,creds,ps)

def runFromFile(content,vm,creds,fileName):
    ''' Execute script file located on CHR '''
    ps = vim.vm.guest.ProcessManager.ProgramSpec(
                programPath = 'import',
                arguments = fileName
    )
    return content.guestOperationsManager.processManager.StartProgramInGuest(vm,creds,ps)


def findDatastore(content,name):
    sessionManager = content.sessionManager

    dcenterObjView = content.viewManager.CreateContainerView(content.rootFolder, [vim.Datacenter], True)

    datacenter = None
    datastore = None
    for dc in dcenterObjView.view:
        dstoreObjView = content.viewManager.CreateContainerView(dc, [vim.Datastore], True)
        for ds in dstoreObjView:
            if ds.info.name == name:
                datacenter = dc
                datastore = ds
                break
        dstoreObjView.Destroy()

    dcenterObjView.Destroy()

    return datacenter,datastore

def _FAILURE(s,*a):
    print(s.format(*a))
    sys.exit(-1)

#------------------------------------------------------------------------------#
```

```
if __name__ == '__main__':
    host = sys.argv[1] # ip or something
    user = 'root'
    pwd = 'MikroTik'
    vmName = 'chr-test'
    dataStoreName = 'datastore1'



    service = connect.SmartConnectNoSSL(host=host,user=user,pwd=pwd)
    if not service:
        _FAILURE("Could not connect to the specified host using specified username and password")

    content = service.RetrieveContent()


    #---------------------------------------------------------------------------
    # Find datacenter and datastore


    datacenter,datastore = findDatastore(content,dataStoreName)

    if not datacenter or not datastore:
        connect.Disconnect(service)
        _FAILURE('Could not find datastore \'{}\'',dataStorename)


    #---------------------------------------------------------------------------
    # Locate vm


    vmxPath = '[{0}] {1}/{1}.vmx'.format(dataStoreName, vmName)
    vm = content.searchIndex.FindByDatastorePath(datacenter, vmxPath)

    if not vm:
        connect.Disconnect(service)
        _FAILURE("Could not locate vm")


    #---------------------------------------------------------------------------
    # Setup credentials from user name and pasword

    creds = vim.vm.guest.NamePasswordAuthentication(username = 'admin', password = '')


    #---------------------------------------------------------------------------
    # Run script

    pm = content.guestOperationsManager.processManager

    try:
        # Run script
        src = [':ip address add address=192.168.0.1/24 interface=ether1;']
        jobID = runInline(content, vm, creds, src)

        # Or run file (from FTP root)
        # jobID = runFromFile(content,vm,creds, 'scripts/provision.rsc')


        #---------------------------------------------------------------------------
        # Wait for job to finish

        pm = content.guestOperationsManager.processManager
        jobInfo = pm.ListProcessesInGuest(vm, creds, [jobID])[0]
        while jobInfo.endTime is None:
            time.sleep(1.0)
            jobInfo = pm.ListProcessesInGuest(vm, creds, [jobID])[0]

        if jobInfo.exitCode != 0:
            _FAILURE('Script failed!')
    except:
        raise
    else:
        connect.Disconnect(service)
```

KVM

QEMU guest agent is available. Supported agent commands can be retrieved by using guest-info command. Host-guest file transfer can be performed by using guest-file-* commands. Guest networking information can be retrieved by using the guest-network-get-interfaces command.

- Scripts can be executed by using the guest-exec command together with the GuestExec data structure:
  - If the *path* member is provided, the corresponding file is executed
  - If the *path* member is not set and *input-data* member is provided, *input-data* value is used as script input
  - If *capture-output* is set, script output is reported back
  - *args* and *env* members are not used

- Script job progress can be monitored with guest-exec-status command. The GuestExecStatus data structure is populated as follows:
  - On success *exitcode* member is set to 0
  - If the script timed out *exitcode* is set to 1
  - If the script contained errors *exitcode* is set to -1
  - *signal* member is not set
  - The *err-data* member is not used
  - If *capture-output* was true, Base64 encoded script output is stored in *out-data*

- An additional agent channel ('chr.provision_channel') is also available