Remote Network Connection™ Installation and Administration Guide for Linux

Remote Network Connection™ Installation and Administration Guide for Linux

Product version: 2.0.1

Documentation version: 5

This document was last updated on: March 28, 2019

Legal Notice

CloudTrust Ltd.

http://cloudtrust.solutions

Technical Support

For information about CloudTrust's support offerings, you can visit our website at the following URL:

http://cloudtrust.solutions/support/

All support services will be delivered in accordance with your support agreement and the then-current enterprise technical support policy.

Contacting Technical Support

Customers with a current support agreement may access Technical Support information at the following URL:

http://cloudtrust.solutions/support/

Before contacting Technical Support, make sure you have satisfied the system requirements that are listed in your product documentation. Also, you should be at the computer on which the problem occurred, in case it is necessary to replicate the problem.

When you contact Technical Support, please have the following information available:

- Product release level
- Hardware information
- Available memory, disk space, and NIC information
- Operating system
- Version and patch level
- Network topology
- Router, gateway, and IP address information
- Problem description:
    - Error messages and log files
    - Troubleshooting that was performed before contacting CloudTrust
    - Recent software configuration changes and network changes

Contents

Installing Remote Network Connection™

Planning the installation

Plan your installation structure

Before you install the product, consider the size and geographical distribution of your network to determine the installation architecture. To ensure good network performance, you need to evaluate several factors. These factors include how many computers need networking, whether any of those computers connect over a wide-area network, or how often to works.

- If your network is small, is located in one geographic location, and has fewer than 500 clients, you need to use only one Remote Network Connection™ server.
- If the network is very large, you can install additional sites. To provide additional redundancy, you can install additional sites for failover or load balancing support.
- If your network is geographically dispersed, you may need to install additional servers for load balancing and bandwidth distribution purposes.

Prepare for and then install Remote Network Connection™

Make sure the computer on which you install the management server meets the minimum system requirements.

To install Remote Network Connection™, you must be logged on with an account that grants local administrator access.

Install Remote Network Connection™.

Decide on the following items when you configure the Remote Network Connection™:

- A Remote Network Connection™ server address, which may be needed depending on the options that you select during installation
- A Ticket ID, which may be needed depending on the options that you select during installation
- A Passphrase, which may be needed depending on the options that you select during installation
- A local internet connection proxy settings, proxy server address, username and password

System requirements for Remote Network Connection™

Component	Requirements
Processor	64-bit processor: 2 GHz Pentium 4 with x86-64 support or equivalent minimum (x86-64 or amd64), ARM (armhf)
Physical RAM	512 MB (1 GB recommended), or higher if required by the operating system
Hard drive	60 MB of available hard disk space for the installation; additional space is required for content and logs
Operating system (server)	Red Hat Enterprise Linux 7 CentOS 7 Oracle Linux 7 Fedora 25, Fedora 26 Debian 8.7 or later versions Ubuntu 17.04, Ubuntu 16.04, Ubuntu 14.04 Linux Mint 18, Linux Mint 17 openSUSE 42.2 or later versions SUSE Enterprise Linux (SLES) 12 SP2 or later versions
Graphical desktop environments	You can use the following graphical desktop environments to view the Remote Network Connection™ for Linux client: • KDE • Gnome • Unity

Remote Network Connection™ client for Linux system requirements

Internationalization requirements

Certain restrictions apply when you install Remote Network Connection™ in a non-English or mixed-language environment.

Component	Requirements
Computer names, server names, and workgroup names	Non-English characters are supported with the following limitations: • Network audit may not work for a host or user that uses a double-byte character set or a high-ASCII character set. • Double-byte character set names or high-ASCII character set names may not appear properly on the Remote Network Connection™ client user interface. • A long double-byte or high-ASCII character set host name cannot be longer than what NetBIOS allows.
English characters	English characters are required in the following situations: • Deploy a client package to a remote computer. • Define the installation path for Remote Network Connection™.

Internationalization requirements

Product license requirements

If you want to use Remote Network Connection™ after the trial period expires, you must purchase and then activate a product license.

A trial license refers to a fully functioning installation of Remote Network Connection™ operating within the free evaluation period. If you want to continue using Remote Network Connection™ beyond the evaluation period, you must purchase and activate a license for your installation. You do not need to uninstall the software to convert from trialware to a licensed installation.

The evaluation period is 1 day from the initial installation of Remote Network Connection™.

Supported virtual installations and virtualization products

You use the Remote Network Connection™ clients to protect the supported operating systems that run in the virtual environments. You can also install and manage Remote Network Connection™ on the supported operating systems that run in virtual environments. You install Remote Network Connection™ on the guest operating system, not the host.

Component	Virtualization product
Remote Network Connection™ components	Windows Azure Amazon WorkSpaces VMware WS 5.0 (workstation) or later VMware GSX 3.2 (enterprise) or later VMware ESX 2.5 (workstation) or later VMware ESXi 4.1 - 5.5 Microsoft Virtual Server 2005 Microsoft Enterprise Desktop Virtualization (MED-V) Windows Server 2008 Hyper-V Windows Server 2012 Hyper-V Windows Server 2012 R2 Hyper-V Citrix XenServer 5.6 or later Virtual Box, supplied by Oracle

Supported virtualization products

Network architecture considerations

You can install Remote Network Connection™ for testing purposes without considering your company network architecture. You can install Remote Network Connection™ with a few clients, and become familiar with the features and functions.

When you are ready to install the production clients, you should plan your deployment based on your organizational structure and computing needs.

You should consider the following elements when you plan your deployment:

- Remote console: Administrators can use a remote computer that runs the console software to access Remote Network Connection™. Administrators may use a remote computer when they are away from the office. You should ensure that remote computers meet the remote console requirements.
- Local and remote computers: Remote computers may have slower network connections. You may want to use a different installation method than the one you use to install to local computers.
- Portable computers such as notebook computers: Portable computers may not connect to the network on a regular schedule.
- Computers that are located in secure areas: Computers that are located in secure areas may need different security settings from the computers that are not located in secure areas.

Communication server ports

Remote Network Connection™ uses the following ports by default. To change the default ports, choose the custom installation type when you run the Setup Wizard.

All communication over these ports is sent over TCP.

Setting	Default	Description
Remote Network Connection™ server port	443	Remote Network Connection™ server listens on this port. The Remote Network Connection™ clients communicate with the server on this port.
Remote Network Connection™ LiveUpdate server port	443	The Remote Network Connection™ clients checks and downloads the latest client version on this port.

Remote Network Connection™ ports

Installing Microsoft .NET Core

To install Microsoft .NET Core

Note: You must have superuser privileges to install the Remote Network Connection™ client on the Linux computer. The procedure uses sudo to demonstrate this elevation of privilege.

To install Microsoft .NET Core for Linux

Supported Linux versions

.NET Core 2.0 treats Linux as a single operating system. There is a single Linux build (per chip architecture) for supported Linux distros. NET Core 2.x is supported on the following Linux 64-bit (x86-64 or amd64) distributions/versions:

- Red Hat Enterprise Linux 7
- CentOS 7
- Oracle Linux 7
- Fedora 25, Fedora 26
- Debian 8.7 or later versions
- Ubuntu 17.04, Ubuntu 16.04, Ubuntu 14.04
- Linux Mint 18, Linux Mint 17
- openSUSE 42.2 or later versions
- SUSE Enterprise Linux (SLES) 12 SP2 or later versions

Prerequisites for .NET Core on Linux

https://docs.microsoft.com/en-us/dotnet/core/linux-prerequisites?tabs=netcore2x

NET Core native installers are available for supported Linux distributions/versions. The native installers require admin (sudo) access to the server. The advantage of using a native installer is that all of the .NET Core native dependencies are installed. Native installers also install the .NET Core SDK system-wide.

Installing Remote Network Connection™

To install Remote Network Connection™

Note: You must have superuser privileges to install the Remote Network Connection™ client on the Linux computer. The procedure uses sudo to demonstrate this elevation of privilege.

To install the Remote Network Connection™ client for Linux

Download the installation package. The package is a .tar.gz file.

```
bash
sudo apt-get install wget
sudo wget https://rnc.services/setup.tar.gz
```

Copy the installation package that you downloaded to the Linux computer. The package is a .tar.gz file.

On the Linux computer, open a terminal application window.

Navigate to the working directory/installation directory with the following command:

```
bash
cd $(pwd)
```

Where directory is the name of the directory into which you copied the .tar.gz file.

Extract the contents of the .tar.gz file into a directory named bin with the following command:

```
bash
mkdir bin
tar -zxvf setup.tar.gz -C ./bin
```

Where setup.tar.gz is the full name of the .tar.gz file, and bin represents a destination folder into which the extraction process places the installation files.

If the destination folder does not exist, the extraction process creates it.

Navigate to bin with the following command:

```
bash
cd bin
```

To correctly set the execute file permissions on install.sh, use the following command:

```
bash
chmod u+x install.sh
```

Use the built-in script to install Remote Network Connection™ with the following command:

```
bash
sudo -i # and change directory to $(pwd)/bin again
./install.sh -i
```

Enter your password if prompted.

This script initiates the installation of the Remote Network Connection™ components. The default installation directory is as follows: *$(pwd)/bin*

The installation completes when the command prompt returns. You do not have to restart the computer to complete the installation.

After the initial installation completes, you configure the Remote Network Connection™ server address and Ticket ID.

```
appsettings.json
{
  "TicketID" :  "",
  "ServerAddress" :  "services.rnc.services:443",
  "NetworkPassphrase" :  "",
  "MemberID" :  "",
  "MemberDescription" :  "sample description"
}
```

* Note: Your Ticket ID should be in an email that shows you bought Remote Network Connection™ subscription. A Remote Network Connection™ Ticket ID is a value consisting of one group of 8 hexadecimal digits, followed by three groups of 4 hexadecimal digits each, followed by one group of 12 hexadecimal digits - groups separated by hyphens. The Ticket ID looks similar to this: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

** Note: A Remote Network Connection™ Member ID is a unique value consisting of one group of 8 hexadecimal digits, followed by three groups of 4 hexadecimal digits each, followed by one group of 12 hexadecimal digits - groups separated by hyphens. The Member ID looks similar to this: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX Universally Unique ID library (uuid-runtime or uuid-devel package) is suitable for generating a unique identifier.

Running Remote Network Connection™ as a Linux daemon

To run Remote Network Connection™ as a Linux daemon

Note: You must have superuser privileges to install the Remote Network Connection™ client on the Linux computer. The procedure uses sudo to demonstrate this elevation of privilege. Linux has a case-sensitive file system.

Create SystemD rncsvc.service file.

rncsvc.service

```
[Unit]
Description=Remote Network Connection Service
After=network.target

[Service]
ExecStart=/usr/bin/dotnet $(pwd)/bin/RNCService.dll
Restart=always
#Restart=on-failure
RestartSec=10 # Restart service after 10 seconds if dotnet service crashes
SyslogIdentifier=remotenetworkconnection

[Install]
WantedBy=multi-user.target
```

*Note: RestartSec parameter must be set correctly. The value of the RestartSec parameter may depend on the number of applications running in the system, or the load and resources of the system. The too low value of the RestartSec parameter may have a negative effect on the operation of the Remote Network Connection system.

Configure SystemD

Copy service file to a System location.

bash

```
sudo cp rncsvc.service /lib/systemd/system
```

Reload SystemD and enable the service, so it will restart on reboots.

bash

```
sudo systemctl daemon-reload
sudo systemctl enable rncsvc
```

Start the service.

bash

```
sudo systemctl start rncsvc
```

Start the service and verify that it's running.

```bash
systemctl status rncsvc
```

Tail the service log.

Remote Network Connection application's output can be examined with journalctl.

```bash
journalctl --unit rncsvc --follow
```

Stopping and restarting the service

Stop the service.

```bash
sudo systemctl stop rncsvc
systemctl status rncsvc
```

Restart the service.

```bash
sudo systemctl start rncsvc
systemctl status rncsvc
```

Cleaning up

Ensure service is stopped.

```
bash
sudo systemctl stop rncsvc
```

Disable.

```
bash
sudo systemctl disable rncsvc
```

Remove and reload SystemD.

```
bash
sudo rm rncsvc.service /lib/systemd/system/rncsvc.service
sudo systemctl daemon-reload
```

Verify SystemD.

```
bash
systemctl --type service |& grep rncsvc
```

Viewing logs

A centralized journal includes all entries for all services and processes managed by systemd. To view the rncsvc.service specific items, use the following command:

```
bash
sudo journalctl -fu rncsvc.service
```

For further filtering, time options such as --since today, --until 1 hour ago or a combination of these can reduce the amount of entries returned.

```
bash
sudo journalctl -fu rncsvc.service --since "2018-10-13" --until "2018-10-13 04:00"
```

Remote Network Connection™ and IP Masquerading

Remote Network Connection™ and Internet Connection Sharing

The purpose of IP Masquerading is to allow machines with private, non-routable IP addresses on your network to access the Internet through the machine doing the masquerading. Traffic from your private network destined for the Internet must be manipulated for replies to be routable back to the machine that made the request. To do this, the kernel must modify the source IP address of each packet so that replies will be routed back to it, rather than to the private IP address that made the request, which is impossible over the Internet.

Linux uses Connection Tracking to keep track of which connections belong to which machines and reroute each return packet accordingly. Traffic leaving your private network is thus "masqueraded" as having originated from your Ubuntu gateway machine. This process is referred to in Microsoft documentation as Internet Connection Sharing.

This can be accomplished with a single iptables rule, which may differ slightly based on your network configuration:

bash
sudo iptables -t nat -A POSTROUTING -s 192.168.0.0/16 -o wlan0 -j MASQUERADE

*Note: without the -s switch, any IP address range to be routed through the specified network device

The above command assumes that your private address space is 192.168.0.0/16 and that your Internet-facing device is wlan0. The syntax is broken down as follows:

- -t nat -- the rule is to go into the nat table
- -A POSTROUTING -- the rule is to be appended (-A) to the POSTROUTING chain
- -s 192.168.0.0/16 -- the rule applies to traffic originating from the specified address space
- -o wlan0 -- the rule applies to traffic scheduled to be routed through the specified network device
- -j MASQUERADE -- traffic matching this rule is to "jump" (-j) to the MASQUERADE target to be manipulated as described above

Each chain in the filter table - the default table, and where most or all packet filtering occurs - has a default policy of ACCEPT, but if you are creating a firewall in addition to a gateway device, you may have set the policies to DROP or REJECT, in which case your masqueraded traffic needs to be allowed through the FORWARD chain for the above rule to work:

```
bash
sudo iptables -A FORWARD -s 192.168.0.0/16 -o wlan0 -j ACCEPT
sudo iptables -A FORWARD -d 192.168.0.0/16 -m state --state ESTABLISHED,RELATED -i wlan0 -j ACCEPT
```

The above commands will allow all connections from your local network to the Internet and all traffic related to those connections to return to the machine that initiated them.

Fine-tune Remote Network Connection™ logging

To fine-tune Remote Network Connection™ logging

Remote Network Connection™ can be configured to make fewer log entries. In this case, journal entries also can be radically reduced - which is especially useful when using hardware (e.g RASPBERRY PI 3 MODEL B, embedded PC) with SD card with limited write cycles.

Stop the service.

bash
sudo systemctl stop rncsvc systemctl status rncsvc

Modify appsettings.json and add RealtimeDebugInformation parameter.

appsettings.json

{
… ,
"RealtimeDebugInformation" : "off"
}

Restart the service.

bash
sudo systemctl start rncsvc systemctl status rncsvc

Applying Remote Network Connection™ interface events

To applying Remote Network Connection™ interface events

Remote Network Connection™ can be configured to generate events for interface events. In this case, information about the up and down events of the interface can be obtained. The interface up event is called asynchronously, the down event is synchronous. This means that the interface up event triggers a command on a separate thread. Of course, the interface up event can be used to run events that are in the interface down event - the command triggered by the interface down event can be left empty.

Stop the service.

bash

```
sudo systemctl stop rncsvc
systemctl status rncsvc
```

Modify appsettings.json and add InterfaceUpEvent and InterfaceDownEvent parameter.

appsettings.json

```
{
  … ,
  "InterfaceUpEvent":  "interfaceupevent.sh",
  "InterfaceDownEvent": "interfacedownevent.sh"
}
```

Create interfaceupevent.sh file.

interfaceupevent.sh

```
#!/bin/bash
echo -e "\033[0;32mRemote Network Connection Interface Up Asynchronous event raised\033[0m"
echo "Starting Service..."
sleep 5 # Sample waiting, only for the purpose of demonstrating synchronous and asynchronous operation
# custom commands
echo " Service started"
echo -e "\033[0;32mRemote Network Connection Interface Up Asynchronous event completed\033[0m"
```

Create interfacedownevent.sh file.

interfacedownevent.sh
#!/bin/bash echo -e "\033[0;32mRemote Network Connection Interface Down Synchronous event raised\033[0m" echo "Stopping Service..." sleep 5 # Sample waiting, only for the purpose of demonstrating synchronous and asynchronous operation # custom commands echo "Service stopped" echo -e "\033[0;32mRemote Network Connection Interface Down Synchronous event completed\033[0m"

* Note: the interface interfacedownevent.sh is initialized only during normal operation. It may happen that the Remote Network Connection service does not stop properly, so the interface interfacedownevent.sh is not called. In this case, the interfaceupevent.sh event can be used to manage related services.

Set execute permissions.

bash
chmod u+x interfaceupevent.sh chmod u+x interfacedownevent.sh

Restart the service.

bash
sudo systemctl start rncsvc systemctl status rncsvc

Parameters that can be processed during an event

The interface up event command (interfaceupevent.sh) gets the list of current members of the Remote Network Connection Network as a parameter (${1}). Processing this list is suitable for accessing each IP addresses of network members in the network. Individual IP addresses can be used as a parameter for separate commands or to create a cache database.

Modify interfaceupevent.sh file.

interfaceupevent.sh

```
#!/bin/bash
echo -e "\033[0;32mRemote Network Connection Interface Up Asynchronous event raised\033[0m"
echo "Starting Service..."


TARGET_LIST=${1}                                #Remote Network Connection Member List
IFS=';' read -ra MEMBERS_LIST <<< "$TARGET_LIST"

for i in "${MEMBERS_LIST[@]}"; do
   MEMBERS=$i
          IFS=' ' read -ra MEMBER <<< "$MEMBERS"
          IP=${MEMBER[3]}
             IFS=':' read -ra IP <<< "$IP"
             #echo ${MEMBER[0]}              #Remote Network Connection Member computer name
             #echo ${MEMBER[1]}              #Remote Network Connection Member ID
             #echo ${MEMBER[2]}              #Remote Network Connection Member description
             echo ${IP[1]}              #Remote Network Connection Member IP address
done


sleep 5 # Sample waiting, only for the purpose of demonstrating synchronous and asynchronous operation
# custom commands
echo " Service started"
echo -e "\033[0;32mRemote Network Connection Interface Up Asynchronous event completed\033[0m"
```

Running multiple Remote Network Connection™ as a Linux daemon

To run multiple Remote Network Connection™ as a Linux daemon

Note: You must have superuser privileges to install the Remote Network Connection™ client on the Linux computer. The procedure uses sudo to demonstrate this elevation of privilege. Linux has a case-sensitive file system.

Make a copy of the previously installed Remote Network Connection™ application into a different directory – eg. bin2.

Create SystemD rncsvc2.service file, and step-by-step to make the previous settings but now with the new directory and the new daemon configuration file.

Modify appsettings.json and add AdapterIndex parameter – default AdapterIndex is 0

appsettings.json

{
… ,
"AdapterIndex" : "1"
}

Start the service.

bash
sudo systemctl start rncsvc2

Add routing tables and rules binding source IP address for each route, and add those as default gateway for each network interface.

Add new routing tables in /etc/iproute2/rt_tables

cat /etc/iproute2/rt_tables
100 t1 101 t2

Add routes to those routing tables

bash
ip route add 10.238.0.0/16 dev rnc0 src 10.238.197.6 table t1 # ip route add table t1 default via 10.238.197.254 dev rnc0 # ip route show table t1 default via 10.238.197.254 dev rnc0 10.238.0.0/16 dev rnc0 scope link src 10.238.197.6

Add routes to those routing tables

```
bash
# ip route add 10.238.0.0/16 dev rnc1 src 10.238.197.7 table t2
# ip route add table t2 default via 10.238.197.254 dev rnc1
# ip route show table t2
default via 10.238.197.254 dev rnc1
10.238.0.0/16 dev rnc1  scope link  src 10.238.197.7
```

Add rules to apply traffic to the routing tables

```
bash
# ip rule add table t1 from 10.238.197.6
# ip rule add table t2 from 10.238.197.7
# ip route show
10.238.0.0/16 dev rnc0  scope link
```

Set interfaces ready for receiving ARP replies

```
bash
# sysctl net.ipv4.conf.default.arp_filter=1
```

To make this routes persistent following configuration files have to be changed

For receiving ARP replies:

```
bash
# grep arp_filter /etc/sysctl.conf
net.ipv4.conf.all.arp_filter = 1
net.ipv4.conf.default.arp_filter = 1
```

For sending ARP:

```
bash
# grep arp_announce /etc/sysctl.conf
net.ipv4.conf.all.arp_announce = 2
net.ipv4.conf.default.arp_announce = 2
```

Checking ping with -I IPADDR

```
bash
# ping -I 10.238.197.6 DSTADDR
# ping -I 10.238.197.7 DSTADDR
```

Product updates

Product updates are improvements to the installed client software. These updates are usually created to extend the operating system or hardware compatibility, adjust performance issues, or fix product errors. Product updates are released on an as-needed basis. Clients can receive product updates directly from a Remote Network Connection™ LiveUpdate server.

Prerequisites for .NET Core on Linux

Linux distribution dependencies

Ubuntu

Ubuntu distributions require the following libraries installed:

libunwind8, liblttng-ust0, libcurl3, libssl1.0.0, libuuid1, libkrb5, zlib1g, libicu52 (for 14.X), libicu55 (for 16.X), libicu57 (for 17.X)

CentOS

CentOS distributions require the following libraries installed:

libunwind, lttng-ust, libcurl, openssl-libs, libuuid, krb5-libs, libicu, zlib

More information: https://docs.microsoft.com/en-us/dotnet/core/linux-prerequisites

Interconnection with other networks

There are several ways to extend the virtual network segment created by the Remote Network Connection™ VPN solution, depending on the usage needs and the requirements of the applications to be used:

- IP Routing,
- Port Forwarding can all be used.

The Remote Network Connection™ VPN solution can be extended using other, classic, standard network solutions - e.g. L2TP/IPSec server connections.

To select the best way to connect to the other networks, you may need to investigate the usage patterns, and you may need to analyze the operation and the traffic patterns of the applications to be used.

The Remote Network Connection™ VPN solution can be used to build point-to-point, point-to-site, and even site-to-site connections.

Remote Network Connection™ by default enables split tunneling

When you configure a Remote Network Connection™ VPN solution, the default setting is to enable split tunneling. What split tunneling refers to is the fact that only connections to the private network are sent over the Remote Network Connection™ VPN tunnels. If the user wants to connect to resources on the Internet, the connection is made over the local link - that is to say, the connection is sent directly to the Internet based on the IP addressing configuration on the Remote Network Connection™ client computer's NIC.

Remote Network Connection™ by default use 10.238.0.0/16 IPv4 addresses as VPN address as a unique IP address to each Remote Network Connection™ Adapter in the range of 10.238.0.0/16 subnet.

What is split tunneling?

Split tunneling is a computer networking concept which allows the user to access dissimilar security domains like a public network (e.g., the Internet) and a local LAN or WAN at the same time, using the same or different network connections. This connection state is usually facilitated through the simultaneous use of a Local Area Network (LAN) Network Interface Card (NIC), Wireless Local Area Network (WLAN) NIC, and Remote Network Connection™ VPN client software application without the benefit of access control.

The advantage of split tunneling is that users have a much better computing experience, especially when accessing Internet based resources. In addition, users on the private network are likely to have a better computing experience when accessing resources on the Internet, since the Remote Network Connection™ client traffic isn't consuming corporate Internet bandwidth to connect to Internet resources - it alleviates bottlenecks and conserves bandwidth. The combined advantages of improved Remote Network Connection™ client and private network client Internet computing experience makes it worthwhile to make split tunneling the default configuration for Remote Network Connection™ VPN communications.

Another advantage is in the case where a user works at a supplier or partner site and needs access to network resources on both networks throughout the day. Split tunneling prevents the user from having to continually connect and disconnect.

Identity authentication with Remote Network Connection™

Remote Network Connection™ server can be configured to require endpoint SSL authentication certificates when establishing the SSL/TLS communications channel. Remote Network Connection™ VPN application supports USB tokens as a mean to increase identity authentication, to enforce security policy while keeping it simple for users and IT managers.

Remote Network Connection™ VPN application is natively interoperable with a large range of tokens and smart cards because it supports smart cards interface to the Microsoft Smart Card Base Cryptographic Service Provider (CSP). Remote Network Connection™ VPN application always accesses the Windows Certificate Store in CSP mode.

Remote Network Connection™ VPN application recognizes the smart cards or USB tokens from leading manufacturers (Gemalto, Oberthur, Aladdin, SafeNet, Yibico, Feitian, etc.). The list below shows the qualified or supported tokens with Remote Network Connection™ VPN application:

- SafeNet/Aladdin - eToken PRO 64k
- Yubico - Yubikey Neo
- Omnikey - Omnikey Cardman 3121, Omnikey CM 6121
- Gemalto - Gemalto IDBridge K30 USB Token

Remote Network Connection™ VPN application can be configured to check the certification chain of the certificate received from the Remote Network Connection™ server. This feature requires importing the root certificate and all the certificates of the certification chain in the Windows Certificate Store.

Remote Network Connection™ VPN application will also use the CRL (Certification Revocation List) of each intermediate certification authority. This CRL must be accessible, either in the Windows Certificate Store or downloadable. If not, the Remote Network Connection™ VPN application won't be able to check the validity of the certificate.

Remote Network Connection™ VPN application checks the following elements of the certification chain:

- the expiration date of the certificate
- validity beginning date of the certificate
- signature of each certificate in the certification chain (included the root certificate, the intermediate certificates, and the server certificate)
- the update of each CRL

* Unique installation of Remote Network Connection™ VPN solution allows integrating Identity authentication into Remote Network Connection™ application's sign-in processes and providing secure TLS communication channel.

Remote Network Connection™ in Docker Container

Docker can build images automatically by reading the instructions from a Dockerfile. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. Using docker build users can create an automated build that executes several command-line instructions in succession.

This paragraph describes the commands you can use in a Dockerfile to install Remote Network Connection™.

Remote Network Connection™ Dockerfile location: https://rnc.services/Dockerfile

Create a Dockerfile for Remote Network Connection™ application

Create a Dockerfile in your project folder.

Add the text below to your Dockerfile for either Linux or Windows Containers. The tags below are multi-arch meaning they pull either Windows or Linux containers depending on what mode is set in Docker Desktop for Windows. Reference: https://docs.docker.com/engine/reference/builder/

Dockerfile

```
# Remote Network Connection(TM) setup for Docker Container
FROM mcr.microsoft.com/dotnet/core/sdk

ENV RNC_TICKET_ID=
ENV RNC_SERVER_ADDRESS=services.rnc.services
ENV RNC_NETWORK_PASSPHRASE=

# Installing required packages
RUN apt-get update
RUN apt-get upgrade -y
RUN apt-get install sudo
RUN apt-get install libunwind8
RUN apt-get install wget
RUN apt-get install openssl
RUN apt-get install uuid-runtime

# Download Remote Network Connection setup.tar.gz package
RUN wget https://rnc.services/setup.tar.gz

# Extracting Remote Network Connection application
RUN mkdir ./root/bin
RUN tar -zxvf setup.tar.gz -C ./root/bin

# Setting up Remote Network Connection appsettings.json parameter file
RUN echo "{" >> ./root/bin/appsettings.json
RUN echo "  \"TicketID\" :  \"$RNC_TICKET_ID\"," >> ./root/bin/appsettings.json
```

```
RUN echo "  \"ServerAddress\" :  \"$RNC_SERVER_ADDRESS:443\"," >> ./root/bin/appsettings.json
RUN echo -n "  \"NetworkPassphrase\" :  \"" >> ./root/bin/appsettings.json
RUN printf $RNC_NETWORK_PASSPHRASE | openssl base64 | awk 'BEGIN{ORS="";} {print}' >> ./root/bin/appsettings.json
RUN echo "\"," >> ./root/bin/appsettings.json
RUN echo -n "  \"MemberID\" :  \"" >> ./root/bin/appsettings.json
RUN echo "" | uuidgen | tr /a-z/ /A-Z/ | tr "\n" "\"," >> ./root/bin/appsettings.json
RUN echo "," >> ./root/bin/appsettings.json
RUN echo "  \"MemberDescription\" :  \"$HOSTNAME\"" >> ./root/bin/appsettings.json
RUN echo "}" >> ./root/bin/appsettings.json

# Finalizing Remote Network Connection setup
RUN chmod u+x ./root/bin/RNCLnxClient.out

# Running Remote Network Connection application
WORKDIR ./root/bin/
ENTRYPOINT ["dotnet", "RNCService.dll"]
```

Build and run the Remote Network Connection™ Docker image

Open a command prompt and navigate to your project folder.

Use the following commands to build and run your Docker image:

```
docker
$ docker build -t rnc:latest .
$ docker run --cap-add=NET_ADMIN --device=/dev/net/tun -it rnc
```

List Docker images

```
docker
$ docker image ls
```

Remove Remote Network Connection™ Docker image

```
docker
$ docker image rm -f rnc
```