Ylianst / MeshAgent Public

Notifications Fork 86 Star 224

Code Issues 84 Pull requests 5 Actions Projects Security Insights

master Code

1,953 Commits

.github/workflows
docs
lib-jpeg-turbo
meshconsole
meshcore
meshreset
meshservice
microscript
microstack
modules
openssl
snippets
test
webrtc
.gitignore
MeshAgent-2022.sln
MeshAgent.sln
clean.bat
compileall
makefile
readme.md

About

MeshAgent used along with MeshCentral to remotely manage computers. Many variations of the background management agent are included as binaries in the MeshCentral project.

meshcentral.com

Readme

Activity

224 stars

22 watching

86 forks

Report repository

Releases

74 tags

Packages

No packages published

Contributors 10

Languages

C 80.0% JavaScript 16.8% C++ 2.0% C# 0.8% Makefile 0.2% HTML 0.2%

README

MeshCentral Agent

Table of Contents

Feedback
License

About

The MeshCentral Agent is the software that runs on remote devices and connects to the MeshCentral server to allow for remote device management. This agent is compiled for Windows, many different Linux distributions, macOS and FreeBSD. In addition compiled for many different processors x86-32, x86-64, ARM, MIPS. For most users, install the MeshCentral server first and install the agent from your server.

To try out this software on the public server, please visit MeshCentral.com/login. Be mindful that the public MeshCentral server comes with no guaranties, most should setup their own server.

For more information, visit MeshCentral.com.

Social Media

Reddit
Twitter
BlogSpot

MSH format

The MeshAgent gets it's security and connection instructions from a .msh text file formatted with each line being a "key=value". The .msh file is generally created by the MeshCentral server and, for example, would look like this:

```
MeshName=MyComputers
MeshType=2
MeshID=0xEDBE1BE...
ServerID=D99362D5...
MeshServer=wss://example.com:443/agent.ashx
```

Here is a list of the possible keys that are currently supported by the agent. Note that the key name must have the exact capitalization:

```
AgentCapabilities           Integer Mask, specifying supported Agen
agentName                   If set, this will be sent to the server
compactDirtyMinimum         Minimum dirty bytes threshold for db.co
controlChannelDebug         If set, will log/display controlChannel
controlChannelIdleTimeout   Integer value specifying the idle timeo
coreDumpEnabled             If set, a dump file will be written whe
disableUpdate               If set, will prevent the agent from sel
noUpdateCoreModule          If set, will prevent the agent from tak
enableILibRemoteLogging     Integer value specifying the port numbe
fakeUpdate                  If set, when the agent self-updates, it
forceUpdate                 If set, will cause the agent to perform
ignoreProxyFile             If set, will cause the agent to ignore
logUpdate                   If set, will cause the agent to log sel
jsDebugPort                 Specify a JS Debugger Port
nocertstore                 If set on Windows, will force the Agent
remoteMouseRender           If set, will always render the remote m
skipmaccheck                If set, the agent will not change NodeI
showModuleNames             If set, will display the name of module
slaveKvmLog                 [Linux] If set, will enable logging ins
WebProxy                    Manually specify proxy configuration
```

Many of these values are used by developers and are not typically used in normal use.

Special notes for BSD systems

You'll need to mount procfs, which isn't mounted by default on FreeBSD. Add the following line to /etc/fstab

```
proc    /proc   procfs  rw      0       0
```

If you don't reboot, then you can manually mount with the command:

```
mount -t procfs proc /proc
```

In addition, it is recommended to install bash, which you can do with the following command:

```
pkg install bash
```

Special Note about KVM Support on Linux:

If you get an error stating that an Xauthority cannot be found, and asking if your DM is configured to use X, or if you get a black screen when connecting to the login screen, you may need to:

- Open /etc/gdm/custom.conf or /etc/gdm3/custom.conf
- Uncomment: WaylandEnable=false.
- Add the following line to the [daemon] section:

```
DefaultSession=gnome-xorg.desktop
```

If you are using LightDM, and the child KVM process unexpectedly closes while connecting to the login screen, you'll need to:

- Open /etc/lightdm/lightdm.conf
- Uncomment the following line from the [LightDM] section:

```
user-authority-in-system-dir=false
```

Special Note For ChromeOS:

You need to disable rootfs verification, in order to install the meshagent service. After running the following commands, and rebooting, you should be able to install the meshagent service.

```
sudo su -
cd /usr/share/vboot/bin/
./make_dev_ssd.sh --remove_rootfs_verification
```

The above line will return a warning, but it will tell you the boot partition number, which you will need when specifying the above command again, this time with the --partions options. Specify the number instead of (ID)

```
./make_dev_ssd.sh --remove_rootfs_verification --partitions ID
reboot
```

When you are ready to install the agent, you'll need to copy the binary to a path that is not marked noexec, like /usr/local, so that you can execute the installer from there.

Self Test

The MeshAgent has the ability to run a self test script, to aid in testing features of the Mesh Agent in consistent/reproducable fashion.

There are two modes of operation of the self test:

- Stand-alone Mode. This runs the tests in place directly from the command line
- IPC Mode. This opens an IPC to a currently running/installed agent, and runs the tests from there.

The easiest way to run the self tests, is by using Stand-alone mode, as it can be quickly setup and run anywhere. To get started, you will need make sure that the agent that you will be testing, has successfully connected to the server at least once. This is becuase the self test will simulate a server connection with the agent, so the agent will need to be running the meshcore that was pushed by the server, which is stored in the db. If the agent has not yet connected with a server, or the meshcore is missing from the .db, the self test will error out with a message stating that the meshcore could not be loaded.

Once the agent has successfully connected at least once, simply exit/stop the agent. Make sure that you have placed a copy of the agent-selftest.js file from the modules folder of the agent repository to the folder where the agent that you wish to test is located.

To start the test on Windows, from a command prompt, simply run the agent from the command line, with the --selfTest=1 switch:

```
MeshAgent --selfTest=1
```

The self test does not explicitly need an elevated command prompt to run, but some tests, such as the AMT tests require elevated permissions.

To start the test on other platforms, simply run the agent from a console session. It will run best from a GUI capable console session, so that it can test features such as user consent. If you use SU to elevate the self test, it is best to pass in the -p flag, to preserve the envionment variables. To start the test, simply run the agent with the --selfTest=1 switch:

```
./MeshAgent --selfTest=1
```

To run the self test in IPC Mode, requires a little more preparation. For security reasons, by default the agent does not allow running 'eval' commands thru the IPC channel, however, the Self Test IPC Mode, requires this functionality. The simplest way to allow this command, is to add the following entry in the msh configuration file of the agent, then restart the agent:

```
debugConsole=1
```

To verify if this flag has been enabled, from the console tab for the agent, run the following command:

```
eval debugConsole
```

If this returns 'true', then you are ready to run the self test in IPC Mode. To start the self test in IPC Mode, start the self test similarly to how you launch the Stand Alone test, from an elevated console, but add the following command line switch:

```
--serviceName="xxx"
```

substituting xxx, with the service name of the agent you are trying to test. If you do not know the service name, you can navigate to the folder that contains the agent, and run the the agent from the command line with the following command line switch:

```
-name
```

This will return the service name for that agent.

After concluding the tests, you can again disable the debugConsole flag by editing the msh configuration file of the agent like the following, and restarting the agent:

```
debugConsole=
```

This will ensure that the debugConsole flag is cleared on the agent. This can be verified by running the following console command, similar to before. It should return false:

```
eval debugConsole
```

Self Test Coverage

The following is the list of basic tests that the Stand Alone test mode will test:

- Server Initialization. This verifies that the agent sends the correct startup sequence to the server.
- AMT/LMS. If detected, will attempt to verify LMS operation.
- Console command. Verifies that the agent is responsive to performing console commands.
- Basic Telemetry. Verifies the operation of some basic telemetry operations requested from the server.
- Web RTC. Verifies ability to establish Peer Data Channels, and transfer large data chunks.
- Non-English UTF8 Dialog Boxes. Linux specific test to verify capability to display non-english utf8 characters on configurable dialog boxes.
- Tunnel Sessions. Verifies ability of agent to establish tunnel session with server.
- Remote Terminal. Verifies ability of agent to establish terminal session with user-consent.
- Remote KVM. Verifies ability of agent to establish KVM session with user-consent, and verifies receipt of correct KVM packets.
- File Transfer Upload. Verifies ability to upload files, by uploading a random stream of bytes.
- File Transfer Download. Verifies ability to doanload files, by downloading the bytes uploaded in previous test, and verifying CRC.

In addition to the above tests, IPC Mode test adds the following tests:

- Mesh Core Dump Test. Attempts to clear/restart JS core while running KVM test, verifying the agent does not crash.
- Service Restart Test. Verifies that the agent can successfully restart itself.

Feedback

If you encounter a problem or have a suggestion to improve the product, you may file an issue report

If you are filing a problem report, you should include:

- The version of the software you are using
- The Operating System and version
- The observed output
- The expected output
- Any troubleshooting you took to resolve the issue yourself
- Any other similar reports

License