
wsdd

wsdd implements a Web Service Discovery host daemon. This enables (Samba) hosts, like your local NAS device, to be found by Web Service Discovery Clients like Windows.

It also implements the client side of the discovery protocol which allows to search for Windows machines and other devices implementing WSD. This mode of operation is called discovery mode.

Purpose

Since NetBIOS discovery is not supported by Windows anymore, wsdd makes hosts to appear in Windows again using the Web Service Discovery method. This is beneficial for devices running Samba, like NAS or file sharing servers on your local network. The discovery mode searches for other WSD servers in the local subnet.

Background

With Windows 10 version 1511, support for SMBv1 and thus NetBIOS device discovery was disabled by default. Depending on the actual edition, later versions of Windows starting from version 1709 (“Fall Creators Update”) do not allow the installation of the SMBv1 client anymore. This causes hosts running Samba not to be listed in the Explorer’s “Network (Neighborhood)” views. While there is no connectivity problem and Samba will still run fine, users might want to have their Samba hosts to be listed by Windows automatically.

You may ask: What about Samba itself, shouldn’t this functionality be included in Samba!? Yes, maybe. However, using Samba as file sharing service is still possible even if the host running Samba is not listed in the Network Neighborhood. You can still connect using the host name (given that name resolution works) or IP address. So you can have network drives and use shared folders as well. In addition, there is a patch lurking around in the Samba bug tracker since 2015. So it may happen that this feature gets integrated into Samba at some time in the future.

Requirements

wsdd requires Python 3.7 and later only. It runs on Linux, FreeBSD, OpenBSD and MacOS. Other Unixes, such as NetBSD, might work as well but were not tested.

Although Samba is not strictly required by wsdd itself, it makes sense to run wsdd only on hosts with a running Samba daemon. Note that the OpenRC/Gentoo init script depends on the Samba service.

Installation

Operating System and Distribution-Depending Instructions

This section provides instructions how to install wsdd on different OS distributions. Sufficient privileges are assumed to be in effect, e.g. by being root or using sudo.

Arch Linux

Install wsdd from the AUR package.

CentOS, Fedora, RHEL

wsdd is included in RedHat/CentOS' EPEL repository. After setting that up, you can install wsdd like on Fedora where it is sufficient to issue

```
1 dnf install wsdd
```

Debian-based Distributions (Debian, Ubuntu, Mint, ...)

Wsdd is included in the official package repositories of Debian and Ubuntu (*universe*) since versions 12 (*Bookworm*) and 22.04 LTS (*Jammy Jellyfish*), respectively. This also applies to Linux Mint, starting from version 21 (*Vanessa*). Thus, it is sufficient to install it via

```
1 apt install wsdd
```

FreeBSD

The wsdd port can be installed via

```
1 pkg install py39-wsdd
```

Gentoo

You can choose between two overlays: the GURU project and an author-maintained dedicated overlay which can be selected as follows

```
1 emerge eselect-repository
2 eselect repository enable guru
3 emerge --sync
```

After setting up one of them you can install wsdd with

```
1 emerge wsdd
```

Generic Installation Instructions

No installation steps are required. Just place the `wsdd.py` file anywhere you want to, rename it to `wsdd`, and run it from there. The init scripts/unit files assume that `wsdd` is installed under `/usr/bin/wsdd` or `/usr/local/bin/wsdd` in case of FreeBSD. There are no configuration files. No special privileges are required to run `wsdd`, so it is advisable to run the service as an unprivileged, possibly dedicated, user for the service.

The `etc` directory of the repo contains sample configuration files for different `init(1)` systems, e.g. FreeBSD's `rc.d`, Gentoo's `openrc`, and `systemd` which is used in most contemporary Linux distros. Those files may be used as templates. They are likely to require adjustments to the actual distribution/installation where they are to be used.

Usage

Firewall Setup

Traffic for the following ports, directions and addresses must be allowed.

- incoming and outgoing traffic to `udp/3702` with multicast destination:
 - `239.255.255.250` for IPv4
 - `ff02::c` for IPv6
- outgoing unicast traffic from `udp/3702`
- incoming to `tcp/5357`

You should further restrict the traffic to the (link-)local subnet, e.g. by using the `fe80::/10` address space for IPv6. Please note that IGMP traffic must be enabled in order to get IPv4 multicast traffic working.

For UFW and `firewalld`, application/service profiles can be found in the respective directories. Note that UFW profiles only allow to grant the traffic on specific UDP and TCP ports, but a restriction on the IP range (like link local for IPv6) or the multicast traffic is not possible.

Options

By default wsdd runs in host mode and binds to all interfaces with only warnings and error messages enabled. In this configuration the host running wsdd is discovered with its configured hostname and belong to a default workgroup. The discovery mode, which allows to search for other WSD-compatible devices must be enabled explicitly. Both modes can be used simultaneously. See below for details.

General options

- `-4, --ipv4only` (see below)
- `-6, --ipv6only`

Restrict to the given address family. If both options are specified no addreses will be available and wsdd will exit.

- `-A, --no-autostart` Do not start networking activities automatically when the program is started. The API interface (see man page) can be used to start and stop the networking activities while the application is running.

- `-c DIRECTORY, --chroot DIRECTORY`

Chroot into a separate directory to prevent access to other directories of the system. This increases security in case of a vulnerability in wsdd. Consider setting the user and group under which wsdd is running by using the `-u` option.

- `-H HOPLIMIT, --hoplimit HOPLIMIT`

Set the hop limit for multicast packets. The default is 1 which should prevent packets from leaving the local network segment.

- `-i INTERFACE/ADDRESS, --interface INTERFACE/ADDRESS`

Specify on which interfaces wsdd will be listening on. If no interfaces are specified, all interfaces are used. The loop-back interface is never used, even when it was explicitly specified. For interfaces with IPv6 addresses, only link-local addresses will be used for announcing the host on the network. This option can be provided multiple times in order to use more than one interface.

This option also accepts IP addresses that the service should bind to. For IPv6, only link local addresses are actually considered as noted above.

- `-l PATH/PORT, --listen PATH/PORT` Enable the API server on the with a Unix domain socket on the given PATH or a local TCP socket bound to the given PORT. Refer to the man page for details on the API.

-
- `--metadata-timeout TIMEOUT` Set the timeout for HTTP-based metadata exchange. Default is 2.0 seconds.

- `-s, --shortlog`

Use a shorter logging format that only includes the level and message. This is useful in cases where the logging mechanism, like systemd on Linux, automatically prepend a date and process name plus ID to the log message.

- `-u USER[:GROUP], --user USER[:GROUP]`

Change user (and group) when running before handling network packets. Together with `-c` this option can be used to increase security if the execution environment, like the init system, cannot ensure this in another way.

- `-U UUID, --uuid UUID`

The WSD specification requires a device to have a unique address that is stable across reboots or changes in networks. In the context of the standard, it is assumed that this is something like a serial number. Wsdd attempts to read the machine ID from `/etc/machine-id` and `/etc/hostid` (in that order) before potentially chrooting in another environment. If reading the machine ID fails, wsdd falls back to a version 5 UUID with the DNS namespace and the host name of the local machine as inputs. Thus, the host name should be stable and not be modified, e.g. by DHCP. However, if you want wsdd to use a specific UUID you can use this option.

- `-v, --verbose`

Additively increase verbosity of the log output. A single occurrence of `-v/--verbose` sets the log level to INFO. More `-v` options set the log level to DEBUG.

- `-V, --version`

Show the version number and exit.

Host Operation Mode

In host mode, the device running wsdd can be discovered by Windows.

- `-d DOMAIN, --domain DOMAIN`

Assume that the host running wsdd joined an ADS domain. This will make wsdd report the host being a domain member. It disables workgroup membership reporting. The (provided) hostname is automatically converted to lower case. Use the `-p` option to change this behavior.

- `-n HOSTNAME, --hostname HOSTNAME`

Override the host name wsdd uses during discovery. By default the machine's host name is used (look at `hostname(1)`). Only the host name part of a possible FQDN will be used in the default case.

- `-o, --no-server`

Disable host operation mode which is enabled by default. The host will not be discovered by WSD clients when this flag is provided.

- `-p, --preserve-case`

Preserve the hostname as it is. Without this option, the hostname is converted as follows. For workgroup environments (see `-w`) the hostname is made upper case by default. Vice versa it is made lower case for usage in domains (see `-d`).

- `-t, --nohttp`

Do not service http requests of the WSD protocol. This option is intended for debugging purposes where another process may handle the Get messages.

- `-w WORKGROUP, --workgroup WORKGROUP`

By default wsdd reports the host is a member of a workgroup rather than a domain (use the `-d/--domain` option to override this). With `-w/--workgroup` the default workgroup name can be changed. The default work group name is WORKGROUP. The (provided) hostname is automatically converted to upper case. Use the `-p` option to change this behavior.

Client / Discovery Operation Mode

This mode allows to search for other WSD-compatible devices.

- `-D, --discovery`

Enable discovery mode to search for other WSD hosts/servers. Found servers are printed to stdout with INFO priority. The server interface (see `-l` option) can be used for a programatic interface. Refer to the man page for details of the API.

Example Usage

- handle traffic on eth0 only, but only with IPv6 addresses

```
wsdd -i eth0 -6
```

or

```
wsdd --interface eth0 --ipv6only
```

-
- set the Workgroup according to smb.conf and be verbose

```
SMB_GROUP=$(grep -i '^\\s*workgroup\\s*='smb.conf | cut -f2 -d= |  
tr -d '[:blank:]')  
  
wsdd -v -w $SMB_GROUP
```

Technical Description

(Read the source for more details)

For each specified (or all) network interfaces, except for the loopback, an UDP multicast socket for message reception, two UDP sockets for replying using unicast as well as sending multicast traffic, and a listening TCP socket are created. This is done for both the IPv4 and the IPv6 address family if not configured otherwise by the command line arguments (see above). Upon startup a *Hello* message is sent. When wsdd terminates due to a SIGTERM signal or keyboard interrupt, a graceful shutdown is performed by sending a *Bye* message. I/O multiplexing is used to handle network traffic of the different sockets within a single process.

Known Issues

Security

wsdd does not implement any security feature, e.g. by using TLS for the http service. This is because wsdd's intended usage is within private, i.e. home, LANs. The *Hello* message contains the host's transport address, i.e. the IP address, which speeds up discovery (avoids *Resolve* message).

In order to increase the security, use the capabilities of the init system or consider the `-u` and `-c` options to drop privileges and chroot.

Usage with NATs

Do not use wsdd on interfaces that are affected by NAT. According to the standard, the *ResolveMatch* messages emitted by wsdd contain the IP address ("transport address" in standard parlance) of the interface(s) the application has been bound to. When such messages are retrieved by a client (Windows hosts, e.g.) they are unlikely to be able to connect to the provided address which has been subject to NAT. To avoid this issue, use the `-i / --interface` option to bind wsdd to interfaces not affected by NAT.

Tunnel/Bridge Interface

If tunnel/bridge interfaces like those created by OpenVPN or Docker exist, they may interfere with wsdd if executed without providing an interface that it should bind to (so it binds to all). In such cases, the wsdd hosts appears after wsdd has been started but it disappears when an update of the Network view in Windows Explorer is forced, either by refreshing the view or by a reboot of the Windows machine. To solve this issue, the interface that is connected to the network on which the host should be announced needs to be specified with the `-i / --interface` option. This prevents the usage of the tunnel/bridge interfaces.

Background: Tunnel/bridge interfaces may cause Resolve requests from Windows hosts to be delivered to wsdd multiple times, i.e. duplicates of such request are created. If wsdd receives such a request first from a tunnel/bridge it uses the transport address (IP address) of that interface and sends the response via unicast. Further duplicates are not processed due to the duplicate message detection which is based on message UUIDs. The Windows host which receives the response appears to detect a mismatch between the transport address in the ResolveMatch message (which is the tunnel/bridge address) and the IP of the sending host/interface (LAN IP, e.g.). Subsequently, the wsdd host is ignored by Windows.

Contributing

Contributions are welcome. Please ensure PEP8 compliance when submitting patches or pull requests. Opposite to PEP8, the maximum number of characters per line is increased to 120.

Licence

The code is licensed under the MIT license.

Acknowledgements

Thanks to Jose M. Prieto and his colleague Tobias Waldvogel who wrote the mentioned patch for Samba to provide WSD and LLMNR support. A look at their patch set made cross-checking the WSD messages easier.

References and Further Reading

Technical Specification

- Web Services Dynamic Discovery
- SOAP-over-UDP (used during multicast)
- MSDN Documentation on Publication Services Data Structure
- MSDN on Windows WSD Compliance
- ...and the standards referenced within the above.

Documentation and Discussion on Windows/WSD

- Microsoft help entry on SMBv1 is not installed by default in Windows 10 Fall Creators Update and Windows Server, version 1709
- Samba WSD and LLMNR support (Samba Bug #11473)
- Discussion at tenforums.com about missing hosts in network Note: Solutions suggest to go back to SMBv1 protocol which is deprecated! Do not follow this advice.

Other stuff

- There is a C implementation of a WSD daemon, named `wsdd2`. This one also includes LLMNR which `wsdd` lacks. However, LLMNR may not be required depending on the actual network/-name resolution setup.