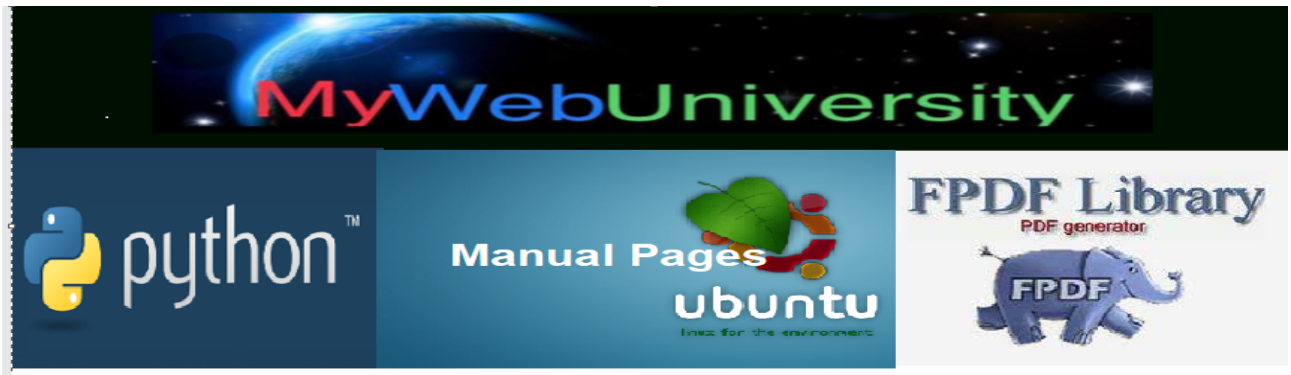

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'varlinkctl.1'

$ man varlinkctl.1

VARLINKCTL(1) varlinkctl VARLINKCTL(1)

NAME

varlinkctl - Introspect with and invoke Varlink services

SYNOPSIS

varlinkctl [OPTIONS...] info ADDRESS

varlinkctl [OPTIONS...] list-interfaces ADDRESS

varlinkctl [OPTIONS...] list-methods ADDRESS [INTERFACE...]

varlinkctl [OPTIONS...] introspect ADDRESS [INTERFACE...]

varlinkctl [OPTIONS...] call ADDRESS METHOD [ARGUMENTS]

varlinkctl [OPTIONS...] --exec call call ADDRESS METHOD ARGUMENTS -- CMDLINE

varlinkctl [OPTIONS...] validate-idl [FILE]

DESCRIPTION

varlinkctl may be used to introspect and invoke Varlink[1] services.

Services are referenced by one of the following:

? A Varlink service reference starting with the "unix:" string, followed by an absolute AF_UNIX socket path, or by "@" and an arbitrary string (the latter for referencing sockets in the abstract namespace). In this case, a stream socket connection is made to the specified socket.

? A Varlink service reference starting with the "exec:" string, followed by an absolute path of a binary to execute. In this case, the specified process is forked off locally, with a connected stream socket passed in.

? A Varlink service reference starting with the "ssh-unix:" string, followed by an SSH host specification, followed by ":", followed by an absolute AF_UNIX socket path. (This

requires OpenSSH 9.4 or newer on the server side, and abstract namespace sockets are not supported.)

? A Varlink service reference starting with the "ssh-exec:" string, followed by an SSH host specification, followed by ":", followed by a command line. In this case, the command is invoked and the Varlink protocol is spoken on the standard input and output of the invoked command.

For convenience, these two simpler (redundant) service address syntaxes are also supported:

? A file system path to an AF_UNIX socket, either absolute (i.e. begins with "/") or relative (in which case it must begin with "./").

? A file system path to an executable, either absolute or relative (as above, must begin with "/" or "./", respectively).

COMMANDS

The following commands are understood:

info ADDRESS

Shows brief information about the specified service, including vendor name and list of implemented interfaces. Expects a service address in one of the formats described above.

Added in version 255.

list-interfaces ADDRESS

Shows a list of interfaces implemented by the specified service. Expects a service address in one of the formats described above.

Added in version 255.

list-methods ADDRESS [INTERFACE...]

Shows a list of methods implemented by the specified service. Expects a service address in one of the formats described above as well as one or more interface names. If no interface name is specified, lists all methods of all interfaces implemented by the service, otherwise just the methods in the specified interfaces.

Added in version 257.

introspect ADDRESS [INTERFACE...]

Shows the interface definitions of the specified interfaces provided by the specified service. Expects a service address in one of the formats described above and optionally one or more Varlink interface names. If no interface names are specified, shows all provided interfaces by the service.

Added in version 255.

call ADDRESS METHOD [ARGUMENTS]

Calls the specified method of the specified service. Expects a service address in the format described above, a fully qualified Varlink method name, and a JSON arguments object. If the arguments object is not specified, it is read from STDIN instead. To pass an empty list of parameters, specify the empty object "{}".

The reply parameters are written as JSON objects to STDOUT.

Added in version 255.

validate-idl [FILE]

Reads a Varlink interface definition file, parses and validates it, then outputs it with syntax highlighting. This checks for syntax and internal consistency of the interface.

Expects a file name to read the interface definition from. If omitted, reads the interface definition from STDIN.

Added in version 255.

help

Shows command syntax help.

Added in version 255.

OPTIONS

The following options are understood:

--more

When used with call: expect multiple method replies. If this flag is set, the method call is sent with the more flag set, which tells the service to generate multiple replies, if needed. The command remains running until the service sends a reply message that indicates it is the last in the series (or if the configured timeout is reached, see below). This flag should be set only for method calls that support this mechanism.

If this mode is enabled, output is automatically switched to JSON-SEQ mode, so that individual reply objects can be easily discerned.

This switch has no effect on the method call timeout applied by default. Regardless of whether --more is specified or not, the default timeout will be 45s. Use --timeout= (see below) to change or disable the timeout. When invoking a method call that continuously returns updates, it is typically desirable to disable the timeout with --timeout=infinity. On the other hand, when invoking a --more method call for the purpose of enumerating objects (which likely will complete quickly), it is typically beneficial to leave the timeout logic enabled, for robustness reasons.

Added in version 255.

-E

A shortcut for --more --timeout=infinity. This switch is useful for method calls that implement subscription to a continuous stream of updates.

Added in version 257.

--collect

This is similar to --more, but collects all responses in a JSON array, and prints it, rather than in JSON-SEQ mode.

Added in version 256.

--oneway

When used with call: do not expect a method reply. If this flag is set, the method call is sent with the oneway flag set (the command exits immediately after), which tells the service not to generate a reply.

Added in version 255.

--json=MODE

Selects the JSON output formatting, either "pretty" for nicely indented, colorized output, or "short" for terse output with minimal whitespace and no newlines. Defaults to "short".

Added in version 255.

-j

Equivalent to --json=pretty when invoked interactively from a terminal. Otherwise, it is equivalent to --json=short, in particular when the output is piped to some other program.

Added in version 255.

--quiet, -q

Suppress output of method call replies.

Added in version 257.

--graceful=

Takes a qualified Varlink error name, i.e. an interface name, suffixed by an error name, separated by a dot, e.g. "org.varlink.service.InvalidParameter". Ensures that, if a method call fails with the specified error, this will be treated as success, i.e. will cause the varlinkctl invocation to exit with a zero exit status. This option may be used more than once in order to treat multiple different errors as successes.

Added in version 257.

--timeout=

Expects a timeout in seconds as parameter. By default, a timeout of 45s is enforced. To turn off the timeout, specify "infinity" or an empty string.

Added in version 257.

--exec

Once the method call issued via call completed successfully, chainload the specified command line, with the method call output parameters serialized to JSON passed into standard input (and standard output and standard error inherited from the invoking process). Moreover any file descriptors passed back on the underlying communication socket are passed to the invoked process via the usual $LISTEN_FDS protocol. This functionality may be used to consume replies that come with associated file descriptors in a reasonable way.

Now that if --exec is specified the the third parameter to call is not optional (i.e. the method call parameters).

Added in version 258.

--push-fd=

Takes a numeric file descriptor number as parameter. May be used to pass a file descriptor along with the method call, if the underlying transport supports this. May be used multiple times to pass multiple file descriptors, retaining the order in which they are specified. The specified file descriptors must be passed to the varlinkctl invocation. Optionally, in place of a numeric file descriptor number an absolute or relative file system path (the latter must be prefixed with "./") may be specified, which is opened in read-only mode.

Added in version 258.

--no-pager

Do not pipe output into a pager.

-h, --help

Print a short help text and exit.

--version

Print a short version string and exit.

EXAMPLES

Example 1. Investigating a Service

The following three commands inspect the "io.systemd.Resolve" service implemented by systemd-resolved.service(8), listing general service information and implemented interfaces, and then displaying the interface definition of its primary interface:

```
$ varlinkctl info /run/systemd/resolve/io.systemd.Resolve
     Vendor: The systemd Project
    Product: systemd (systemd-resolved)
    Version: 254 (254-1522-g4790521^)
        URL: https://systemd.io/
 Interfaces: io.systemd
             io.systemd.Resolve
             org.varlink.service
$ varlinkctl list-interfaces /run/systemd/resolve/io.systemd.Resolve
io.systemd
io.systemd.Resolve
org.varlink.service
$ varlinkctl introspect /run/systemd/resolve/io.systemd.Resolve io.systemd.Resolve
interface io.systemd.Resolve
type ResolvedAddress(
        ifindex: ?int,
        ...
```

(Interface definition has been truncated in the example above, in the interest of brevity.)

Example 2. Invoking a Method

The following command resolves a hostname via systemd-resolved.service(8)'s ResolveHostname method call.

```
$ varlinkctl call /run/systemd/resolve/io.systemd.Resolve io.systemd.Resolve.ResolveHostname '{"name":"systemd.io","family":2}' -j
{
        "addresses" : [
                {
                        "ifindex" : 2,
                        "family" : 2,
                        "address" : [
                                185,
```

```
                199,
                111,
                153
            ]
        }
    ],
    "name" : "systemd.io",
    "flags" : 1048577
}
```

Example 3. Investigating a Service Executable

The following command inspects the /usr/lib/systemd/systemd-pcrextend executable and the IPC APIs it provides. It then invokes a method on it:

```
# varlinkctl info /usr/lib/systemd/systemd-pcrextend
     Vendor: The systemd Project
    Product: systemd (systemd-pcrextend)
    Version: 254 (254-1536-g97734fb)
        URL: https://systemd.io/
Interfaces: io.systemd
            io.systemd.PCRExtend
            org.varlink.service
# varlinkctl introspect /usr/lib/systemd/systemd-pcrextend io.systemd.PCRExtend
interface io.systemd.PCRExtend
method Extend(
        pcr: int,
        text: ?string,
        data: ?string
) -> ()
# varlinkctl call /usr/lib/systemd/systemd-pcrextend io.systemd.PCRExtend.Extend '{"pcr":15,"text":"foobar"}'
{}
```

Example 4. Invoking a method remotely via SSH

The following command acquires a report about the identity of a remote host "somehost" from systemd-hostnamed.service(8) by connecting via SSH to the AF_UNIX socket the service listens on:

```
# varlinkctl call ssh-unix:somehost:/run/systemd/io.systemd.Hostname io.systemd.Hostname.Describe '{}'
```

To invoke a Varlink service binary directly on the remote host, rather than talking to a service via AF_UNIX can be done like this:

```
# varlinkctl call ssh-exec:somehost:systemd-creds org.varlink.service.GetInfo '{}'
```

SEE ALSO

busctl(1), Varlink[1]

NOTES

1. Varlink

   https://varlink.org/