NAME

busctl − Introspect the bus

SYNOPSIS

busctl [OPTIONS...] [COMMAND] [*NAME*...]

DESCRIPTION

busctl may be used to introspect and monitor the D−Bus bus.

OPTIONS

The following options are understood:

−−address=*ADDRESS*
Connect to the bus specified by *ADDRESS* instead of using suitable defaults for either the system or user bus (see **−−system** and **−−user** options).

−−show−machine
When showing the list of peers, show a column containing the names of containers they belong to. See **systemd-machined.service**(8).

−−unique
When showing the list of peers, show only "unique" names (of the form ":*number*.*number*").

−−acquired
The opposite of **−−unique** — only "well−known" names will be shown.

−−activatable
When showing the list of peers, show only peers which have actually not been activated yet, but may be started automatically if accessed.

−−match=*MATCH*
When showing messages being exchanged, show only the subset matching *MATCH*. See **sd_bus_add_match**(3).

−−size=
When used with the **capture** command, specifies the maximum bus message size to capture ("snaplen"). Defaults to 4096 bytes.

−−list
When used with the **tree** command, shows a flat list of object paths instead of a tree.

−q, **−−quiet**
When used with the **call** command, suppresses display of the response message payload. Note that even if this option is specified, errors returned will still be printed and the tool will indicate success or failure with the process exit code.

−−verbose
When used with the **call** or **get−property** command, shows output in a more verbose format.

−−json=*MODE*
When used with the **call** or **get−property** command, shows output formatted as JSON. Expects one of "short" (for the shortest possible output without any redundant whitespace or line breaks) or "pretty" (for a pretty version of the same, with indentation and line breaks). Note that transformation from D−Bus marshalling to JSON is done in a loss−less way, which means type information is embedded into the JSON object tree.

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

−−expect−reply=*BOOL*
When used with the **call** command, specifies whether **busctl** shall wait for completion of the method call, output the returned method response data, and return success or failure via the process exit code. If this is set to "no", the method call will be issued but no response is expected, the tool terminates

immediately, and thus no response can be shown, and no success or failure is returned via the exit code. To only suppress output of the reply message payload, use **−−quiet** above. Defaults to "yes".

−−auto−start=*BOOL*
When used with the **call** or **emit** command, specifies whether the method call should implicitly activate the called service, should it not be running yet but is configured to be auto−started. Defaults to "yes".

−−allow−interactive−authorization=*BOOL*
When used with the **call** command, specifies whether the services may enforce interactive authorization while executing the operation, if the security policy is configured for this. Defaults to "yes".

−−timeout=*SECS*
When used with the **call** command, specifies the maximum time to wait for method call completion. If no time unit is specified, assumes seconds. The usual other units are understood, too (ms, us, s, min, h, d, w, month, y). Note that this timeout does not apply if **−−expect−reply=no** is used, as the tool does not wait for any reply message then. When not specified or when set to 0, the default of "25s" is assumed.

−−augment−creds=*BOOL*
Controls whether credential data reported by **list** or **status** shall be augmented with data from /proc. When this is turned on, the data shown is possibly inconsistent, as the data read from /proc might be more recent than the rest of the credential information. Defaults to "yes".

−−watch−bind=*BOOL*
Controls whether to wait for the specified **AF_UNIX** bus socket to appear in the file system before connecting to it. Defaults to off. When enabled, the tool will watch the file system until the socket is created and then connect to it.

−−destination=*SERVICE*
Takes a service name. When used with the **emit** command, a signal is emitted to the specified service.

−−user
Talk to the service manager of the calling user, rather than the service manager of the system.

−−system
Talk to the service manager of the system. This is the implied default.

−H, **−−host=**
Execute the operation remotely. Specify a hostname, or a username and hostname separated by "@", to connect to. The hostname may optionally be suffixed by a port ssh is listening on, seperated by ":", and then a container name, separated by "/", which connects directly to a specific container on the specified host. This will use SSH to talk to the remote machine manager instance. Container names may be enumerated with **machinectl −H** *HOST*. Put IPv6 addresses in brackets.

−M, **−−machine=**
Execute operation on a local container. Specify a container name to connect to.

−−no−pager
Do not pipe output into a pager.

−−no−legend
Do not print the legend, i.e. column headers and the footer with hints.

−h, **−−help**
Print a short help text and exit.

−−version
Print a short version string and exit.

COMMANDS

The following commands are understood:

list
Show all peers on the bus, by their service names. By default, shows both unique and well−known names, but this may be changed with the **−−unique** and **−−acquired** switches. This is the default operation if no command is specified.

status [*SERVICE*]
Show process information and credentials of a bus service (if one is specified by its unique or well−known name), a process (if one is specified by its numeric PID), or the owner of the bus (if no parameter is specified).

monitor [*SERVICE*...]
Dump messages being exchanged. If *SERVICE* is specified, show messages to or from this peer, identified by its well−known or unique name. Otherwise, show all messages on the bus. Use Ctrl+C to terminate the dump.

capture [*SERVICE*...]
Similar to **monitor** but writes the output in pcap format (for details, see the **Libpcap File Format**[1] description). Make sure to redirect standard output to a file. Tools like **wireshark**(1) may be used to dissect and view the resulting files.

tree [*SERVICE*...]
Shows an object tree of one or more services. If *SERVICE* is specified, show object tree of the specified services only. Otherwise, show all object trees of all services on the bus that acquired at least one well−known name.

introspect *SERVICE OBJECT* [*INTERFACE*]
Show interfaces, methods, properties and signals of the specified object (identified by its path) on the specified service. If the interface argument is passed, the output is limited to members of the specified interface.

call *SERVICE OBJECT INTERFACE METHOD* [*SIGNATURE* [*ARGUMENT*...]]
Invoke a method and show the response. Takes a service name, object path, interface name and method name. If parameters shall be passed to the method call, a signature string is required, followed by the arguments, individually formatted as strings. For details on the formatting used, see below. To suppress output of the returned data, use the **−−quiet** option.

emit *OBJECT INTERFACE SIGNAL* [*SIGNATURE* [*ARGUMENT*...]]
Emit a signal. Takes a object path, interface name and method name. If parameters shall be passed, a signature string is required, followed by the arguments, individually formatted as strings. For details on the formatting used, see below. To specify the destination of the signal, use the **−−destination=** option.

get−property *SERVICE OBJECT INTERFACE PROPERTY*...
Retrieve the current value of one or more object properties. Takes a service name, object path, interface name and property name. Multiple properties may be specified at once, in which case their values will be shown one after the other, separated by newlines. The output is, by default, in terse format. Use **−−verbose** for a more elaborate output format.

set−property *SERVICE OBJECT INTERFACE PROPERTY SIGNATURE ARGUMENT*...
Set the current value of an object property. Takes a service name, object path, interface name, property name, property signature, followed by a list of parameters formatted as strings.

help
Show command syntax help.

PARAMETER FORMATTING

The **call** and **set−property** commands take a signature string followed by a list of parameters formatted as string (for details on D−Bus signature strings, see the **Type system chapter of the D−Bus specification**[2]). For simple types, each parameter following the signature should simply be the parameter's

value formatted as string. Positive boolean values may be formatted as "true", "yes", "on", or "1"; negative boolean values may be specified as "false", "no", "off", or "0". For arrays, a numeric argument for the number of entries followed by the entries shall be specified. For variants, the signature of the contents shall be specified, followed by the contents. For dictionaries and structs, the contents of them shall be directly specified.

For example,

s jawoll

is the formatting of a single string "jawoll".

as 3 hello world foobar

is the formatting of a string array with three entries, "hello", "world" and "foobar".

a{sv} 3 One s Eins Two u 2 Yes b true

is the formatting of a dictionary array that maps strings to variants, consisting of three entries. The string "One" is assigned the string "Eins". The string "Two" is assigned the 32−bit unsigned integer 2. The string "Yes" is assigned a positive boolean.

Note that the **call**, **get−property**, **introspect** commands will also generate output in this format for the returned data. Since this format is sometimes too terse to be easily understood, the **call** and **get−property** commands may generate a more verbose, multi−line output when passed the **−−verbose** option.

EXAMPLES

Example 1. Write and Read a Property

The following two commands first write a property and then read it back. The property is found on the "/org/freedesktop/systemd1" object of the "org.freedesktop.systemd1" service. The name of the property is "LogLevel" on the "org.freedesktop.systemd1.Manager" interface. The property contains a single string:

```
# busctl set−property org.freedesktop.systemd1 /org/freedesktop/systemd1 org.freedesktop.systemd1.Manager LogLevel s
# busctl get−property org.freedesktop.systemd1 /org/freedesktop/systemd1 org.freedesktop.systemd1.Manager LogLevel
s "debug"
```

Example 2. Terse and Verbose Output

The following two commands read a property that contains an array of strings, and first show it in terse format, followed by verbose format:

```
$ busctl get−property org.freedesktop.systemd1 /org/freedesktop/systemd1 org.freedesktop.systemd1.Manager Environme
as 2 "LANG=en_US.UTF−8" "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin"
$ busctl get−property −−verbose org.freedesktop.systemd1 /org/freedesktop/systemd1 org.freedesktop.systemd1.Manager
ARRAY "s" {
        STRING "LANG=en_US.UTF−8";
        STRING "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin";
};
```

Example 3. Invoking a Method

The following command invokes the "StartUnit" method on the "org.freedesktop.systemd1.Manager" interface of the "/org/freedesktop/systemd1" object of the "org.freedesktop.systemd1" service, and passes it two strings "cups.service" and "replace". As a result of the method call, a single object path parameter is received and shown:

```
# busctl call org.freedesktop.systemd1 /org/freedesktop/systemd1 org.freedesktop.systemd1.Manager StartUnit ss "cups.se
o "/org/freedesktop/systemd1/job/42684"
```

SEE ALSO

dbus-daemon(1), **D–Bus**[3], **sd-bus**(3), **systemd**(1), **machinectl**(1), **wireshark**(1)

NOTES

1. Libpcap File Format
   https://wiki.wireshark.org/Development/LibpcapFileFormat
2. Type system chapter of the D-Bus specification
   http://dbus.freedesktop.org/doc/dbus-specification.html#type-system
3. D-Bus
   https://www.freedesktop.org/wiki/Software/dbus