# Application Debugging

Tommy Hagberg

# Confd applications

- Dataproviders (transforms/external DB)
- Hook callbacks
- Validation callbacks
- Actions/Rpc callbacks
- Authorization callbacks
- CDB subscribers
- HA clients
- Event notification subscribers
- CDB/MAAPI clients
- Authentication cb/app

# Confd applications

- Dataproviders (transforms/external DB)
- Hook callbacks
- Validation callbacks
- Actions/Rpc callbacks
- Authorization callbacks
- CDB subscribers
- HA clients
- Event notification subscribers
- CDB/MAAPI clients
- Authentication cb/app

Callback daemons

# Confd applications

- Dataproviders (transforms/external DB)
- Hook callbacks
- Validation callbacks
- Actions/Rpc callbacks
- Authorization callbacks
- CDB subscribers
- HA clients
- Event notification subscribers
- CDB/MAAPI clients
- Authentication cb/app

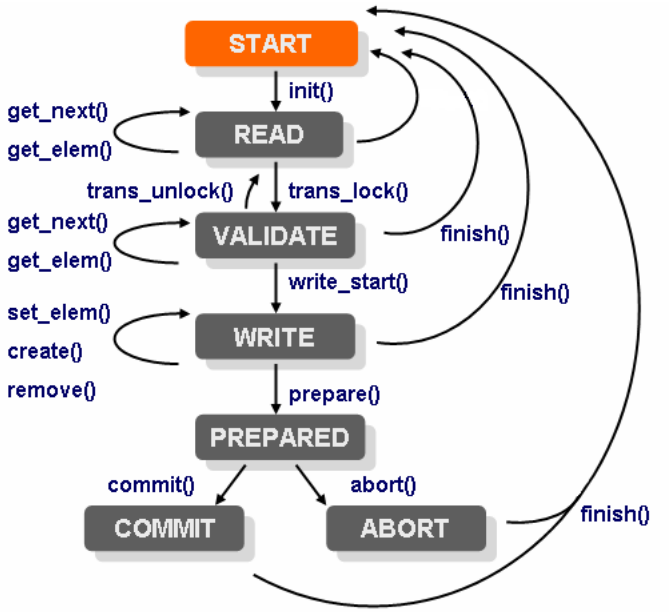
Callback daemons

Devel.log

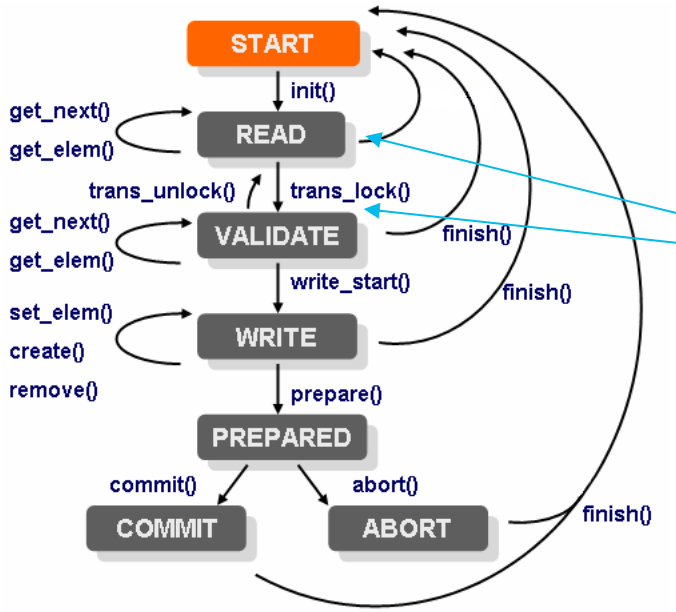
Libconfd trace

Audit.log

# Transaction state

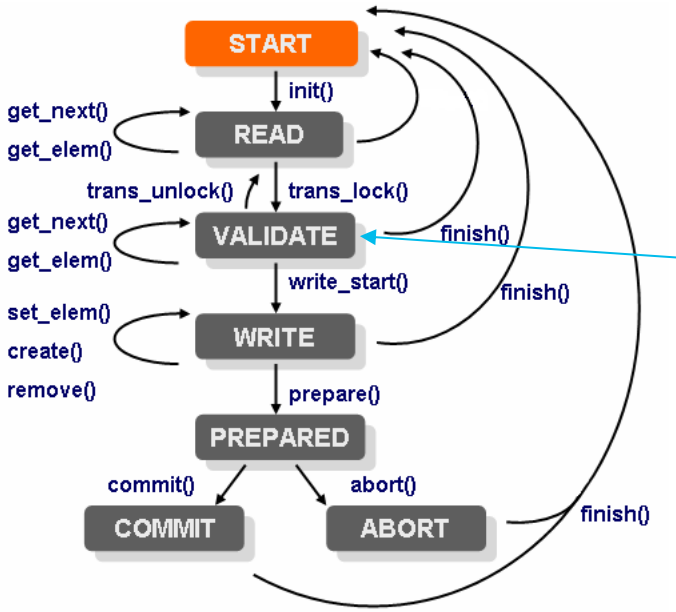


# Transaction states



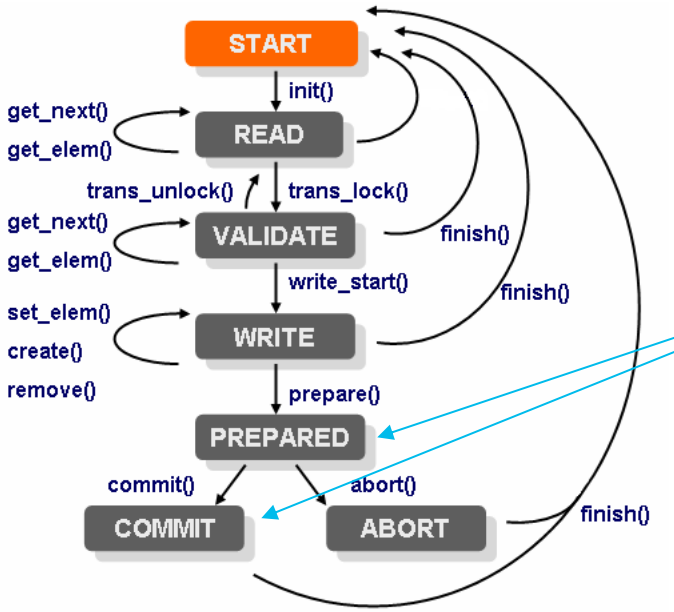

- Hooks

# Transaction states



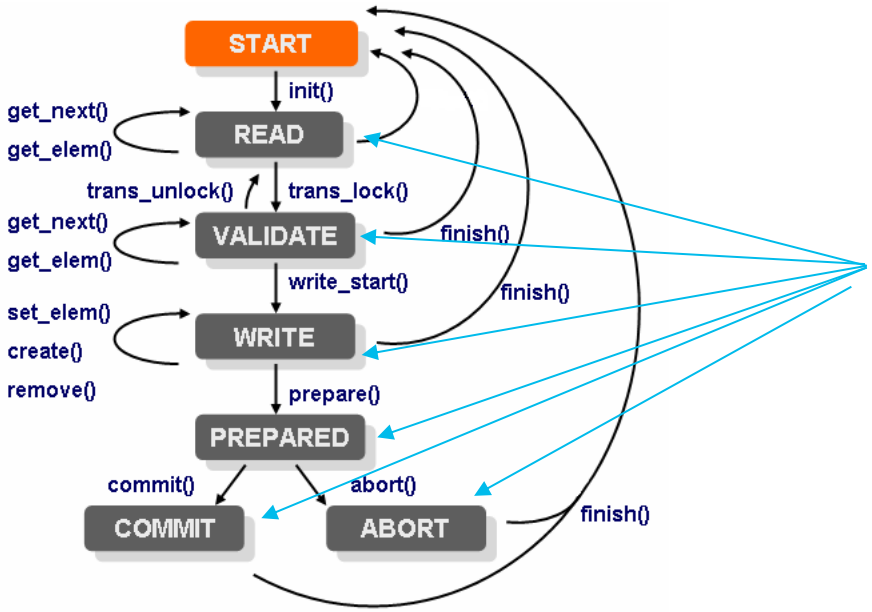

- Validation callbacks

# Transaction states




- CDB subscribers

# Transaction states




- Dataproviders (transforms/external DB)

# Mentioning locks

- Transaction locks, acquired by the transaction engine during validate and commit.
- CDB locks, acquired by CDB subscribers and explicit read sessions.
- Operational CDB data locks, different from CDB lock, only applies to individual operational data leaves.
- Implicit locks, for example, there can only be one concurrent action per user session

# Know the header files

$CONFD_DIR/include/*.h

confd_lib.h

```
/* various values of confd_errno */
#define CONFD_ERR_NOEXISTS          1
#define CONFD_ERR_ALREADY_EXISTS    2
#define CONFD_ERR_ACCESS_DENIED     3
…

/* error codes from apply and validate */
#define CONFD_ERR_NOTSET            12
#define CONFD_ERR_NON_UNIQUE        13
#define CONFD_ERR_BAD_KEYREF        14
…

/* ha related errors */
#define CONFD_ERR_HA_CONNECT        25
#define CONFD_ERR_HA_CLOSED         26
#define CONFD_ERR_HA_BADFXS         27
…
```

# Know the header files

## $CONFD_DIR/include/*.h

confd_lib.h

```
/* various values of confd_errno */
#define CONFD_ERR_NOEXISTS          1
#define CONFD_ERR_ALREADY_EXISTS    2
#define CONFD_ERR_ACCESS_DENIED     3
...

/* error codes from apply and validate */
#define CONFD_ERR_NOTSET            12
#define CONFD_ERR_NON_UNIQUE        13
#define CONFD_ERR_BAD_KEYREF        14
...

/* ha related errors */
#define CONFD_ERR_HA_CONNECT        25
#define CONFD_ERR_HA_CLOSED         26
#define CONFD_ERR_HA_BADFXS         27
...
```

## Also listed in the confd_lib_lib manpage:

ERRORS

...

char *confd_strerror(int code);

returns a string which describes a particular error code. When one of the The following error codes are available:

CONFD_ERR_NOEXISTS (1)

Typically we tried to read a value through CDB or MAAPI which does not exist.

CONFD_ERR_ALREADY_EXISTS (2)

We tried to create something which already exists.

CONFD_ERR_ACCESS_DENIED (3)

Access to an object was denied due to AAA authorization rules.

...

# Know the header files

$CONFD_DIR/include/*.h

confd_lib.h

```
/* various values of confd_errno */
#define CONFD_ERR_NOEXISTS          1
#define CONFD_ERR_ALREADY_EXISTS    2
#define CONFD_ERR_ACCESS_DENIED     3
...

/* error codes from apply and validate */
#define CONFD_ERR_NOTSET            12
#define CONFD_ERR_NON_UNIQUE        13
#define CONFD_ERR_BAD_KEYREF        14
...

/* ha related errors */
#define CONFD_ERR_HA_CONNECT        25
#define CONFD_ERR_HA_CLOSED         26
#define CONFD_ERR_HA_BADFXS         27
...
```

Error handling:

```
result = maapi_get_elem(maapi_socket, tctx->thandle, &v,  firstLoc);

  if ((result != CONFD_OK) && (confd_errno == CONFD_ERR_BADPATH)) {
     LOG("Trying %s.", secondLoc);
     result = maapi_get_elem(maapi_socket, tctx->thandle, &v, secondLoc);
  }

  if ((result != CONFD_OK) && (confd_errno == CONFD_ERR_NOEXISTS)) {
        confd_data_reply_not_found(tctx);
        return CONFD_OK;
```

# Confd Logs

- Northbound logs and traces
  audit.log, netconf.log, netconf.trace etc.
- Confd.log
  Confd daemon log. Consulted e.g for startup problems.
- Devel.log
  Troubleshoot confd application communications (confds' side of the socket) progress, aaa rules, HA etc.
- Libconfd trace
  Troubleshoot confd applications (app's side of the socket)
- Confderr.log
  Logs confd (internal) crashes such as unhandled return values etc. Mainly useful to confd developers

# Debugging verbosity levels

## devel.log

In confd.conf(5)

```
<logs>
   <developerLogLevel>trace</developerLogLevel>
</logs>
```

Levels: error | info | trace

## libconfd trace

In code:

```
void confd_init(const char *name, FILE *estream,
                const enum confd_debug_level debug);

void confd_set_debug(enum confd_debug_level debug,
                     FILE *estream);

enum confd_debug_level {
    CONFD_SILENT = 0,
    CONFD_DEBUG = 1,
    CONFD_TRACE = 2,
    CONFD_PROTO_TRACE = 3
};
```

# Example

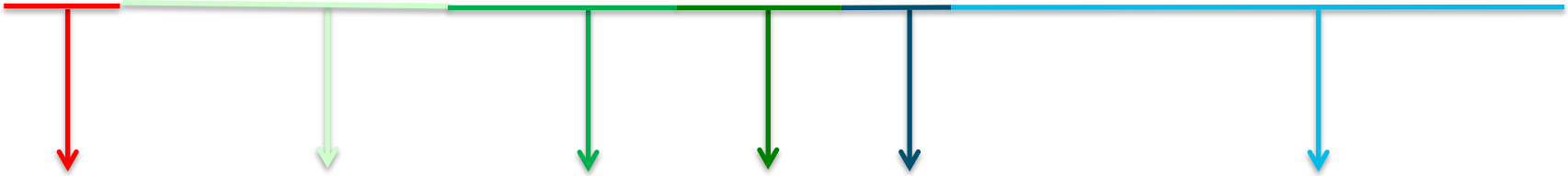
- Example set devel.log level from cli
- Example set libconfd level from cli

# Devel.log format

<DEBUG> 4-Apr-2019::01:59:03.044 TOHAGBER-M-K0CT confd[48146]: confd commit progress db=running usid=30 thandle=21: entering validate phase for running...

...

<DEBUG> 4-Apr-2019::01:59:53.781 TOHAGBER-M-K0CT confd[48146]: devel-c get_next request for callpoint hcp path /hst:host



ALERT
CRIT
ERR
WARNING
NOTICE
INFO
DEBUG

Date

hostname

Confd OS pid

devel-c
devel-cdb
devel-aaa
devel-snmpgw
devel-snmpa
devel-econfd
confd commit progress
...

Message

# Libconfd.trace format

15-Oct-2019::15:01:41.465 24568/7fff9ecf1380/4 GOT {28,2,0,9,arpe,1,101,[],13,0,[[417671722,[1561036191|230872026]],-1,0]}

TRACE CALL data get_next(thandle=9, /arpentries/arpe, -1)

15-Oct-2019::15:01:41.466 24568/7fff9ecf1380/4 SEND {28,2,0,{{0,140729973334128},{{10,147,40,1},#Bin<en0>}}}
--> CONFD_OK

# Libconfd.trace format

```
15-Oct-2019::15:01:41.465 24568/7fff9ecf1380/4 GOT {28,2,0,9,arpe,1,101,[],13,0,[[417671722,[1561036191|230872026]],-1,0]}

TRACE CALL data get_next(thandle=9, /arpentries/arpe, -1)

15-Oct-2019::15:01:41.466 24568/7fff9ecf1380/4 SEND {28,2,0,{{0,140729973334128},{{10,147,40,1},#Bin<en0>}}}
--> CONFD_OK
```

What information is embedded in proto_trace?

# Libconfd.trace format (Received)

15-Oct-2019::15:01:41.465 24568/7fff9ecf1380/4 GOT {28,2,0,9,arpe,1,101,[],13,0,[[417671722,[1561036191|230872026]],-1,0]}

Date

pid/thread/socket

{trans_call, query_ref, daemon_id, thandle, cp, int, data_call,[], usid, 0, [data]}

```
%%- trans callback
-define(CONFD_PROTO_NEW_TRANS,            20).
-define(CONFD_PROTO_TRANS_LOCK,           21).
-define(CONFD_PROTO_TRANS_UNLOCK,         22).
-define(CONFD_PROTO_WRITE_START,          23).
-define(CONFD_PROTO_PREPARE,              24).
-define(CONFD_PROTO_ABORT,                25).
-define(CONFD_PROTO_COMMIT,               26).
-define(CONFD_PROTO_CLOSE_TRANS,          27).
-define(CONFD_PROTO_CALLBACK,             28).
-define(CONFD_PROTO_CALLBACK_TIMEOUT, 29).
-define(CONFD_PROTO_INTERRUPT,            30).
```

```
%%- data callbacks
-define(CONFD_DATA_CB_GET_NEXT,           101).
-define(CONFD_DATA_CB_GET_ELEM,           102).
-define(CONFD_DATA_CB_GET_OBJECT,         103)
...
%%- validate callbacks
-define(CONFD_PROTO_NEW_VALIDATE,         145).
-define(CONFD_PROTO_CLOSE_VALIDATE,       146).
-define(CONFD_VALIDATE_VALUE,             147).
...
%%- action callbacks
-define(CONFD_PROTO_NEW_ACTION,           150).
-define(CONFD_PROTO_ABORT_ACTION,         151).
-define(CONFD_CALL_ACTION,                152).
-define(CONFD_CALL_ACTION_COMMAND,        153).
-define(CONFD_CALL_ACTION_COMPLETION,     154).
...
```

# Libconfd.trace format (Received)

15-Oct-2019::15:01:41.465 24568/7fff9ecf1380/4 GOT {28,2,0,9,arpe,1,101,[],13,0,[[417671722,[1561036191|230872026]],-1,0]}

Date

pid/thread/socket

{trans_call, query_ref, daemon_id, thandle, cp, int, data_call,[], usid, 0, [data]}

```
%%- trans callback
-define(CONFD_PROTO_NEW_TRANS,            20).
-define(CONFD_PROTO_TRANS_LOCK,           21).
-define(CONFD_PROTO_TRANS_UNLOCK,         22).
-define(CONFD_PROTO_WRITE_START,          23).
-define(CONFD_PROTO_PREPARE,              24).
-define(CONFD_PROTO_ABORT,                25).
-define(CONFD_PROTO_COMMIT,               26).
-define(CONFD_PROTO_CLOSE_TRANS,          27).
-define(CONFD_PROTO_CALLBACK,             28).
-define(CONFD_PROTO_CALLBACK_TIMEOUT, 29).
-define(CONFD_PROTO_INTERRUPT,            30).
```

```
%%- data callbacks
-define(CONFD_DATA_CB_GET_NEXT,           101).
-define(CONFD_DATA_CB_GET_ELEM,           102).
-define(CONFD_DATA_CB_GET_OBJECT,         103)
...
%%- validate callbacks
-define(CONFD_PROTO_NEW_VALIDATE,         145).
-define(CONFD_PROTO_CLOSE_VALIDATE,       146).
-define(CONFD_VALIDATE_VALUE,             147).
...
%%- action callbacks
-define(CONFD_PROTO_NEW_ACTION,           150).
-define(CONFD_PROTO_ABORT_ACTION,         151).
-define(CONFD_CALL_ACTION,                152).
-define(CONFD_CALL_ACTION_COMMAND,        153).
-define(CONFD_CALL_ACTION_COMPLETION      154).
...
```

libconfd/src/confd_proto.h

# Libconfd.trace format (SEND)

TRACE CALL data get_next(thandle=9, /arpentries/arpe, -1)

15-Oct-2019::15:01:41.466 24568/7fff9ecf1380/4 SEND {28,2,0,{{0,140729973334128},{{10,147,40,1},#Bin<en0>}}}

Date

pid/thread/socket

{trans_call, counter, }

```
static int get_next(struct confd_trans_ctx *tctx,
                    confd_hkeypath_t *keypath,
                    long next)
  ...
  CONFD_SET_IPV4(&v[0], curr->ip4);
  CONFD_SET_STR(&v[1], curr->iface);
  confd_data_reply_next_key(tctx, &v[0], 2, (long)curr->next);

  return CONFD_OK;
}
```

# Libconfd logger hook

Libconfd provides a hook point for which a library user can install their own log printer. This done by assigning to a global variable **confd_user_log_hook**.

```
void mylogger(int syslogprio, const char *fmt, va_list ap) {
   char buf[BUFSIZ];
   sprintf(buf, "MYLOG:(%d) ", syslogprio);
   strcat(buf, fmt);
   vfprintf(stderr, buf, ap);
 }
…
confd_user_log_hook = mylogger;
```

# Socket timeouts

/confdConfig/capi/newSessionTimeout (xs:duration) [PT30S]

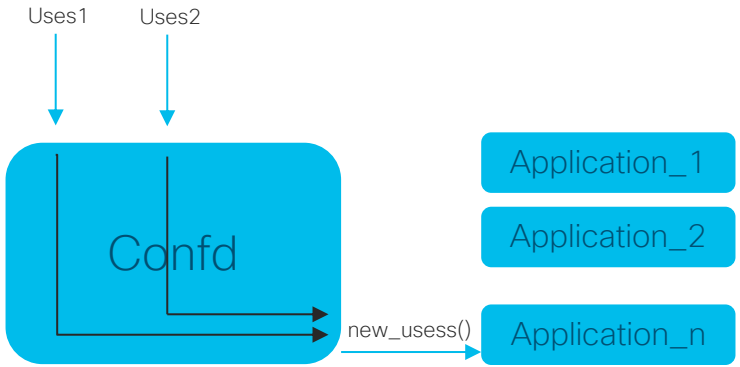
Timeout for a daemon to respond to a control socket request, see confd_lib_dp(3). If the daemon fails to respond within the given time it will be disconnected.

/confdConfig/capi/queryTimeout (xs:duration) [PT120S]

Timeout for a daemon to respond to a worker socket query, see confd_lib_dp(3). If the daemon fails to respond within the given time, it will be disconnected.

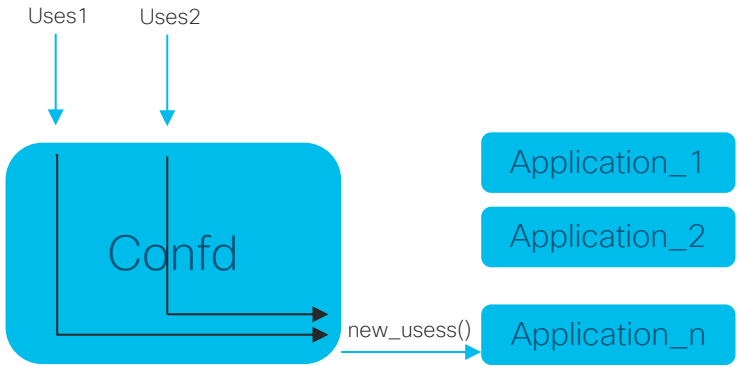
# Example1 ctrl socket timeout

Both session interacts with the dataprovider



# Example1 ctrl socket timeout

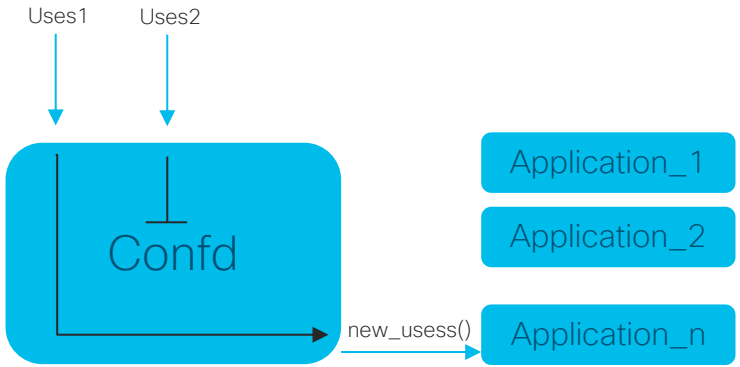
Both session interacts with the dataprovider




Application needs to respond to this new_usess() request within the ctrl-socket timeout (default 30 sec)

# Example2 ctrl socket timeout

Only one session interacts with the dataprovider




Application needs to respond to this new_usess() request within the ctrl-socket timeout (default 30 sec)

# ctrl socket timeout

How do you avoid this?

# Threads

If one have two threads/processes that reads/writes from/to the same socket this can happen:

1. Thread A sends a request and expect a certain response.
2. Thread B does the same.
3. Thread A reads the integer that tells it how many bytes to read.
4. Thread B does the same but get the first 4 bytes of the reply body intended for process A.
5. Thread A reads the number of bytes it should, but the first 4 bytes has already been consumed by B so it reads the first 4 bytes of the next message.

When this happens, three different results can be observed:

1. The 4 bytes makes up a very large integer that's not possible to allocate memory for
2. The 4 bytes makes up a large integer which is possible to allocate memory for, but since ConfD will not send that many bytes the client will be sitting their waiting.
3. The 4 bytes makes up an integer that's smaller than the payload of the message from ConfD the client will get the data and will start to "unpack" it, which will most likely fail since it's not the data it expected.

# Self contained example

- Utilize existing confd examples
- If creating a reproduction example try to use confd utilitie programs e.g confd_cmd, see examples examples.confd/scripting
- Try to create a template setup, that you can reuse. -example
- If difficult to create a self contained example make sure you provide all the logs with correct verbosity levels and make sure those logs capture the time of reproduction