7.7 - The Tcl API

- The Tcl API is similar to the Shell APIs
- All the functions provided by the API are in `TET_ROOT/lib/tcl/tetapi.tcl`

Interface to the user-written test code

- You tell the TCM about the invocable component names in your test case by defining an array called `iclist`
- You tell the TCM the names of your startup and cleanup functions by defining variables called `tet_startup` and `tet_cleanup`

`iclist` - list of invocable component names

- You should initialise each element in the `iclist` array to the name of an invocable component
- Each of these names should have the prefix `ic` followed by the invocable component number
- For example:

```
Set iclist=(ic1,ic2,ic3)
```

- You should define an array for each invocable component, then initialise each element in the array to names of that IC's test purpose functions
- For example:

```
set ic1 "test1"
set ic2 "test2"
set ic3 "test3 test4"
```


`tet_startup` and `tet_cleanup`

- You should set these variables to the names of your startup and cleanup functions

```
set test_startup = startup
set test_cleanup = cleanup
```

- If your test case doesn't use one of these functions you should set the corresponding variable to an empty string

```
set test_startup “”
set test_cleanup “”
```

Tcl API functions and variables

- Making journal entries
  - `tet_infoline "`*`text`*`"`
  - `tet_result` *`result-name`*
  - `tet_setcontext` and `tet_setblock`
- Canceling test purposes
  - `tet_delete` *`test-name`*`[ "`*`reason-string`*`"]`
  - *`deletion-reason`* `= tet_reason "`*`test-name`*`"`
- Name of the current test purpose
  - `$tet_thistest`

Accessing configuration variables

- The API makes configuration variables available to functions as variables within the tcl namespace
- For example, if you define a configuration variable called `MY_VAR`, you would access it in a function as `$MY_VAR`

API library files

- TCM
  - `$TET_ROOT/lib/tcl/tcl.tcm.dat`
- API library
  - `$TET_ROOT/lib/tcl/tclapi.tcl`
- The Tcl TCM must be *sourced* into the test case script by using the `source` command
- This command should be the last line in the file
- For example:

```
source $env{TET_ROOT}/lib/tcl/tcl.tcm.dat
```

- Sourcing the TCM automatically sources the API as well

THE Open GROUP

Building the tcl Binding

- Need to have installed a supported version of TET and tclX
- tclX is needed for support of Signals
- Set TET_ROOT in the environment
- Check the settings of parameters in the Makefile
  - In general if you have installed TET you would not need to make any change since the key parameters are in the high level defines.mk. You may need to change TCL_NSIG
- Build the tcl binding using the make utility

THE Open GROUP

Building the tcl Binding

$ cd contrib/tclapi

```
$make
sed -e 's/STD_SIGNAL_LIST/1 2 3 4 6 8 13 14 15 10 12 20 18
   21 22/' \
    -e 's/SPEC_SIGNAL_LIST/9 17 19 11/' \
    -e 's/TET_NSIG_NUM/32/' \
    tcl.tcm.dat > ../../lib/tcl/tcl.tcm.dat
cp tetapi.tcl ../../lib/tcl
chmod 755 ../../lib/tcl/tcl.tcm.dat ../../lib/tcl/tetapi.tcl
```


Using the tcl Binding

- Ensure that your test code invokes the tcl interpreter in the first line

  ```
  #!/usr/bin/tcl
  ```

- Define the tet_startup and tet_cleanup functions (optional)
- Define the `iclist` array
- Define the individual invocable components referring to test purpose functions

Using the tcl binding

- Define the startup and cleanup procedures referenced by tet_startup and tet_cleanup (optional)
- Define the test purpose functions
- Source the Tcl TCM into the test case script as the last line
- Invoke tcc to run the test cases

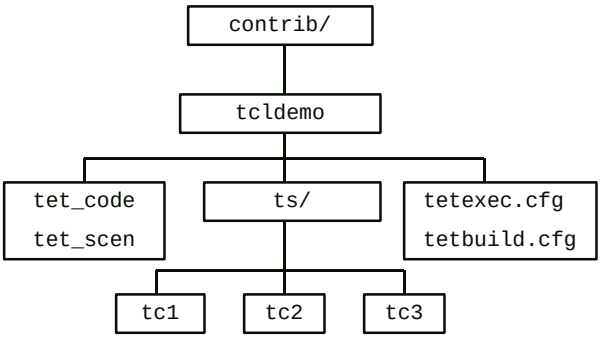
Example Tcl test case

```
#!/usr/bin/tcl
set tet_startup "startup"
set tet_cleanup ""
set iclist "ic1"
set ic1 "tp1"

proc startup {} {
    tet_infoline "Entering the startup routine"
# startup routines typically check for dependencies
# and can cancel tests using tet_delete
    set TET_TCL_DEPEND ""
    if {([catch {set TET_TCL_DEPEND $env(TET_TCL_DEPEND)}] == 1) ||
    ([string length TET_TCL_DEPEND] == 0)} {
        tet_delete tp3 "tp3 deleted since TET_TCL_DEPEND not set in the environment"
    }
}
proc tp1 {} {
    tet_infoline "This is an infoline message from tcl"
    tet_result PASS
}
# execute tcl test case manager - must be last line
source $env(TET_ROOT)/lib/tcl/tcl.tcm.dat
```

The tcldemo test suite

- tcldemo is an example test suite



Invoke tcc to run the tcldemo test suite

- Ensure that you have TET_ROOT defined in your environment
- Assuming the tcldemo is installed under TET_ROOT/contrib, change to the directory and invoke tcc:
  - tcc –p –b contrib/demo
  - tcc -p –e contrib/tcldemo

Running the tcldemo test suite

```
$ tcc -b -p contrib/tcldemo
tcc: journal file is …/tcldemo/results/0005b/journal
09:00:29  Build   /ts/test1.tcl
09:00:30  Build   /ts/test2.tcl
09:00:31  Build   /ts/test3.tcl
$ tcc -e -p contrib/tcldemo
tcc: journal file is …/tcldemo/results/0006e/journal
09:00:35  Execute /ts/test1.tcl
09:00:36  Execute /ts/test2.tcl
09:00:37  Execute /ts/test3.tcl
```