DEVEL package
MATLAB Coding Framework

Laurent Royer

Environment mode

Following the DTAP standard:
- Development
- Testing
- User Acceptance Test
- Production

https://en.wikipedia.org/wiki/Development,_testing,_acceptance_and_production

If pcoded, then the environment mode cannot be DEV
If mode is DEV, then the software cannot be pcoded

Syntax:
>> mode = devel.Env.getMode( )
>> devel.Env.setMode(mode)

Environment mode	Description
DEV	Software in development with MATLAB editor.
TEST	Testing framework.
UAT	Software tested by the end users.
PROD	Software validated and deployed, with increased performances.

Logging level

Objectives:
- Control the amount of information displayed during the execution of the tool, via a log level.
- Improve the performance of the deployed code by removing some internal checks.

The log level can take the following values: OFF / FATAL / ERROR / WARN / INFO / DEBUG / TRACE
https://en.wikipedia.org/wiki/Log4j

This table shows the effect of the Log command according to the environment mode:

DTAP environment	DEV Development	TEST Testing	UAT User Acceptance Test	PROD Production
Source code	mfiles only	mfiles or pcode	mfiles or pcode	mfiles or pcode
devel.Log.fatal(cond,msg)	enter debug mode	throw error	throw error	ignored
devel.Log.error(msg)	enter debug mode	throw error	display error + confirm	ignored
devel.Log.warn(msg)	display warning	display warning	display warning	display warning
devel.Log.info(msg)	display message	display message	display message	display message
devel.Log.debug(msg)	display message	ignored	ignored	ignored
devel.Log.trace(tag,msg)	store message	ignored	ignored	ignored
default log level	TRACE	INFO	INFO	WARN
minimal log level	WARN	INFO	WARN	OFF

Log.fatal syntax:
>> devel.Log.fatal(@() (condition), message)
The condition won't be evaluated in PROD mode

Syntax:
>> devel.Log.fatal(condition handle, single message)
>> devel.Log.error(message)
>> devel.Log.warn(message)
>> devel.Log.info(message)
>> devel.Log.debug(message)
>> devel.Log.trace(message)

Log Level	Description
OFF(0)	Turn off logging.
FATAL(1)	Severe errors that cause premature termination.
ERROR(2)	Other runtime errors or unexpected conditions.
WARN(3)	Use of deprecated APIs, poor use of API, 'almost' errors, other runtime situations that are undesirable or unexpected, but not necessarily "wrong".
INFO(4)	Interesting runtime events (startup/shutdown).
DEBUG(5)	Detailed information on the flow through the system.
TRACE(6)	Most detailed information, written to logs only.

Example 1: If the log level is WARN, then all FATAL, ERROR and WARN messages are enabled

Example 2: In UAT mode, the minimal log level is WARN. So the log level cannot be set to OFF, FATAL or ERROR.

Example 3: In PROD mode, only WARN and INFO message can be displayed.