How to Run Feature Tests when the Game is Already Running?

Feature tests can be run on any non-release build of the game. If you run feature tests by typing into the in-game console, failures are reported in the game log and (if the **Designer Warning** dialog box is supported on the build and enabled) in a pop-up dialog box.

First of all, you'll need to load a file with feature tests in. Do this in the in-game console by typing **ft_load** followed by a valid feature test filename (without the path or filename extension) e.g. **ft_load FeatureTests** or **ft_load CrashSiteTests**. This will load the file into memory but won't start any tests running.

With a file loaded, you can start running tests by typing **ft_runAll** followed by the name of a set of tests defined in the file (or run multiple sets by typing names with a '+' between them). Type **ft_runAll** without any parameters to see a list of all the sets you've loaded... for example, typing **ft_runAll** with the **FeatureTests** file open currently prints out the following:

```
[FEATURETESTER] Please specify which feature test set(s) to run, e.g.
[FEATURETESTER]   ft_runAll utils
[FEATURETESTER]   ft_runAll multiplayerGameServer
[FEATURETESTER]   ft_runAll multiplayerGameClient
[FEATURETESTER]   ft_runAll soloTests
[FEATURETESTER]   ft_runAll sprintTests
[FEATURETESTER]   ft_runAll grenadeTests
[FEATURETESTER]   ft_runAll localScoreTests
[FEATURETESTER]   ft_runAll pickupTests
[FEATURETESTER]   ft_runAll utils+multiplayerGameServer
```

Having told the **Feature Tester** to run a bundle of tests you should then see the game performing the relevant actions and checking that everything's working as it should be.

You'll see the name of each test appear at the bottom of the screen as it's being run; once no name's displayed here, the tests are finished.

If the **Designer Warning** dialog box is enabled you'll be told of failures as and when they're discovered; otherwise you'll have to check the game log to find out about any failures.

How to Run Feature Tests from the Command Line?

The simplest cases are the ones which require only one player. In other situations, you'll need to get two copies of the game running simultaneously. Here are some useful examples (you'll probably want to paste each of these blocks of text into an empty text file and then save it with .bat at the end instead of .txt).

```
Bin32\GameSDK.exe -devmode +autotest_enabled 1 +autotest_state_setup "state:
ATEST_STATE_TEST_FEATURES#outputName:soloTestsInTestMap#file:FeatureTests#rules:TIA#level:basic_testmap_mp#set:
soloTests+localScoreTests+grenadeTests+sprintTests"
```

```
Bin32\GameSDK.exe -devmode +autotest_enabled 1 +autotest_state_setup "state:ATEST_STATE_TEST_FEATURES!file:
CrashSiteTests!outputName:crashSite!rules:CS!level:cw2_rooftop_gardens!set:setup+crashsite"
```

```
start /b Bin32\GameSDK.exe +pl_velocityInterpAlwaysSnap 1 +g_playerEnableDedicatedInput 0 -devmode -logfile
twoPlayersTestLevel_server.log +autotest_enabled 1 +autotest_state_setup "state:ATEST_STATE_TEST_FEATURES!
outputName:twoPlayerInteractionNew!file:TwoPlayers!rules:TIA!level:basic_testmap_mp!set:waitForInit+runOnServer"
choice /T 25 /C ync /CS /D y > nul
start /b Bin32\DedicatedServer.exe -noprompt +pl_velocityInterpAlwaysSnap 1 +g_playerEnableDedicatedInput 0 -
devmode -logfile twoPlayersTestLevel_client.log +autotest_enabled 1 +autotest_state_setup "state:
ATEST_STATE_TEST_FEATURES!outputName:twoPlayerInteractionNew!file:TwoPlayers!set:runOnClient"
```

```
start /b Bin32\GameSDK.exe +pl_velocityInterpAlwaysSnap 1 +g_playerEnableDedicatedInput 0 -devmode -logfile
moreTestServer.log +autotest_enabled 1 +autotest_state_setup "state:ATEST_STATE_TEST_FEATURES!outputName:
killCamTestsServer!file:KillCamTests!rules:TIA!level:basic_testmap_mp!set:runOnServer"
choice /T 25 /C ync /CS /D y > nul
start /b Bin32\GameSDK.exe -noprompt +pl_velocityInterpAlwaysSnap 1 +g_playerEnableDedicatedInput 0 -devmode -
logfile moreTestClient.log +autotest_enabled 1 +autotest_state_setup "state:ATEST_STATE_TEST_FEATURES!
outputName:killCamTestsClient!file:KillCamTests!set:runOnClient"
```

What If I Can't Stick Things On The Command Line All That Easily?

If, for any reason, you'd rather modify your **user.cfg** or **system.cfg** file than add feature-test-executing instructions to the command line, then you can - at least for the tests which require only one player in the game.

For example, dump the following block of text into either one of the aforementioned files and then activate the appropriate line (i.e. remove the preceding ';' character) as required.

```
autotest_enabled = 1
;autotest_state_setup = "state:ATEST_STATE_TEST_FEATURES!file:CrashSiteTests!outputName:crashSite!rules:
CrashSite!level:cw2_rooftop_gardens!set:setup+crashsite"
;autotest_state_setup = "state:ATEST_STATE_TEST_FEATURES!file:FeatureTests!outputName:soloInTestMapTIA!rules:
TIA!level:basic_testmap!set:soloTests+grenadeTests+localScoreTests+sprintTests"
;autotest_state_setup = "state:ATEST_STATE_TEST_FEATURES!file:FeatureTests!outputName:soloInPierTIA!rules:TIA!
level:cw2_pier!set:soloTests+localScoreTests+grenadeTests"
;autotest_state_setup = "state:ATEST_STATE_TEST_FEATURES!file:FeatureTests!outputName:soloInRooftopGardensTIA!
rules:TIA!level:cw2_rooftop_gardens!set:soloTests+localScoreTests+grenadeTests"
;autotest_state_setup = "state:ATEST_STATE_TEST_FEATURES!file:FeatureTests!outputName:soloInCrashTrailCTF!rules:
CTF!level:cw2_alien_crash_trail!set:soloTests+grenadeTests+localScoreTests"
;autotest_state_setup = "state:ATEST_STATE_TEST_FEATURES!file:FrontEndTests!outputName:frontEnd!set:
frontEndTests"
```

Checking The Results Which You've Generated By Running Feature Tests Locally

If you've run tests by setting the **autotest_state_setup** value the game will generate one or more XML report files containing your results in a folder called, appropriately enough, "Xml-Reports". You can open these files in a text editor or a browser.

Each test which was run will have generated a **<testcase>** section in the file. If a test failed then its **<testcase>** section will contain details of the failure (i. e. be followed immediately by a slightly indented **<failure>** section).

Here's what a successful test generates:

```
<testcase name="soloTests: ConsoleKillCommand - The console 'kill' command kills the player" owners="timf"
time="0.10241579" status="run"/>
```

Here's what a failure generates:

```
<testcase name="runOnServer: GrenadeTest - Can throw a grenade at a targetted enemy and kill him" owners="alexh;
claire" time="16.395742">
  <failure type="TestCaseFailed" message="Reached an explicit &apos;Fail&apos; command in the list of test
instructions&#10;FPS = 220.635, level = &apos;!Testmaps/basic_testmap_mp&apos; (TeamInstantAction)&#10;Player
&apos;nottinghambuild&apos; (team=1 &apos;tan&apos;, client, alive, health=120/120, stance=stand, inAir=0.00,
LTag:LTagSingle 7/7+8 LTagGrenade) pos=&lt;1800.00 500.00 268.00&gt; dir=&lt;0.00 1.00 0.00&gt;&#10;Player
&apos;nottinghambuild(1)&apos; (team=2 &apos;black&apos;, alive, health=120/120, stance=stand, inAir=0.00, SCAR:
Rapid 40/40+80 ScarBullet) pos=&lt;1800.00 520.00 268.00&gt; dir=&lt;0.94 -0.35 0.00&gt;&#10;
kFTC_RunFeatureTest (&quot;UtilPlace20mApart&quot;)&#10;    kFTC_RunFeatureTest (&quot;UtilLocalStand&quot;)
&#10;    kFTC_SetItem (&quot;FragGrenades&quot;)&#10;    kFTC_SetResponseToHittingCCCPoint (&quot;
Throw_Prime&quot;, kFTCHR_expectedNext)&#10;    kFTC_SetResponseToHittingCCCPoint (&quot;Throw_Start&quot;,
kFTCHR_expectedNext)&#10;    kFTC_OverrideButtonInput_Press (&quot;attack1_xi&quot;)&#10;
kFTC_WaitUntilHitAllExpectedCCCPoints (0.500000)&#10;    kFTC_SetResponseToHittingCCCPoint (&quot;
Throw_Stop&quot;, kFTCHR_expectedNext)&#10;    kFTC_OverrideButtonInput_Release (&quot;attack1_xi&quot;)
&#10;    kFTC_WaitUntilHitAllExpectedCCCPoints (0.500000)&#10;    kFTC_SetResponseToHittingCCCPoint (&quot;
PlayerState_NonLocalPlayerDied&quot;, kFTCHR_completeTest)&#10;    kFTC_Wait (5.000000)&#10;&gt;   kFTC_Fail ()
&#10;GrenadeTest: &apos;Throw_Prime&apos; has been hit once&#10;GrenadeTest: &apos;Throw_Start&apos; has been
hit once&#10;GrenadeTest: &apos;Throw_Stop&apos; has been hit once&#10;GrenadeTest: &apos;
PlayerState_NonLocalPlayerDied&apos; has never been hit"/>
</testcase>
```

Granted, the message isn't the most readable thing in the world in this format, but **the first sentence of the message will give you the reason why the test failed** and if you want to see the message in all its glory, stick the text into a text editor and do a **Find and Replace** on all the following things (you can normally open this window by hitting Ctrl-H):

Find this...	Replace with this...
\r\n
'	'

&quot;	"
&gt;	>
&lt;	<

Doing all that turns the above message into the following:

```
Reached an explicit 'Fail' command in the list of test instructions
FPS = 220.635, level = '\!Testmaps/basic_testmap_mp' (TeamInstantAction)
Player 'nottinghambuild' (team=1 'tan', client, alive, health=120/120, stance=stand, inAir=0.00, LTag:
LTagSingle 7/7+8 LTagGrenade) pos=<1800.00 500.00 268.00> dir=<0.00 1.00 0.00>
Player 'nottinghambuild(1)' (team=2 'black', alive, health=120/120, stance=stand, inAir=0.00, SCAR:Rapid 40
/40+80 ScarBullet) pos=<1800.00 520.00 268.00> dir=<0.94 \-0.35 0.00>
kFTC_RunFeatureTest ("UtilPlace20mApart")
kFTC_RunFeatureTest ("UtilLocalStand")
kFTC_SetItem ("FragGrenades")
kFTC_SetResponseToHittingCCCPoint ("Throw_Prime", kFTCHR_expectedNext)
kFTC_SetResponseToHittingCCCPoint ("Throw_Start", kFTCHR_expectedNext)
kFTC_OverrideButtonInput_Press ("attack1_xi")
kFTC_WaitUntilHitAllExpectedCCCPoints (0.500000)
kFTC_SetResponseToHittingCCCPoint ("Throw_Stop", kFTCHR_expectedNext)
kFTC_OverrideButtonInput_Release ("attack1_xi")
kFTC_WaitUntilHitAllExpectedCCCPoints (0.500000)
kFTC_SetResponseToHittingCCCPoint ("PlayerState_NonLocalPlayerDied", kFTCHR_completeTest)
kFTC_Wait (5.000000)
>   kFTC_Fail ()
GrenadeTest: 'Throw_Prime' has been hit once
GrenadeTest: 'Throw_Start' has been hit once
GrenadeTest: 'Throw_Stop' has been hit once
GrenadeTest: 'PlayerState_NonLocalPlayerDied' has never been hit
```