# CTF Test Instruction Reference

| Associated Plug-Ins | CTF Instructions | Data/Argument Name & Description | Instruction Usage/Application | Examples |
|---|---|---|---|---|
| cfs | ***RegisterCfs*** | ***target*** **:** name of cFS instance; i.e. name of the executable cFS image; this name should match a section in the .ini file in the format [<***target***>] within the section commented as "Base settings for cfs"; default is an empty string; if not specified, it applies to all cFS targets in the .ini file whose name begins with "cfs_". | Register a cFS target with the specified name so CTF can manage its external interfaces. CTF can manage multiple cFS targets. | `{`<br>`    "instruction":  "RegisterCfs",`<br>`    "data": {`<br>`        "target": "lx_cfs_1"`<br>`    }`<br>`}` |
| cfs | ***BuildCfs*** | ***target:*** name of a registered cFS target; optional; default is empty string; if not specified, it applies to all registered cFS targets | Have CTF build the cFS executable image with the specified name | `{`<br>`    "instruction":  "BuildCfs",`<br>`    "data": {`<br>`        "target": "lx_cfs_1"`<br>`    }`<br>`}` |
| cfs | ***StartCfs*** | ***target:*** name of a registered cFS target; optional; default is empty string; if not specified, it applies to all registered cFS targets<br>***run_args:*** command line argument for cFS executable | Have CTF start up the cFS target with the specified name and arguments | `{`<br>`    "instruction":  "StartCfs",`<br>`    "data": {`<br>`        "target": "lx_cfs_1",`<br>`        "run_args": "-R P0"`<br>`    }`<br>`}` |
| cfs | ***EnableCfsOutput*** | ***target:*** name of a registered cFS target; optional; default is empty string; if not specified, it applies to all registered cFS targets | Have CTF establish the connection to cFS target for telemetry downlink | `{`<br>`    "instruction":  "EnableCfsOutput",`<br>`    "data": {`<br>`        "target": "lx_cfs_1"`<br>`    }`<br>`}` |
| cfs | ***SendCfsCommand*** | ***target:*** name of a registered cFS target; optional; default is empty string; if not specified, it applies to all registered cFS targets<br>***mid:*** CCSDS message ID; can either be a value or its associated name, as defined in the MIDS JSON data file<br>***cc:*** CCSDS command/function code; can either be a value or its associated name, as defined in the CMD JSON data file<br>***args:*** list of command arguments, as defined in the CMD JSON data file<br>***header*** : Optional; object where the key is the header field name, and the value is the field value. Fields depend on the CCSDS header implementation. | Have CTF send a command to the specified target(s) | `{`<br>`    "instruction":  "SendCfsCommand",`<br>`    "data": {`<br>`        "target": "lx_cfs_1",`<br>`        "mid": "TO_CMD_MID",`<br>`        "cc": "TO_ENABLE_OUTPUT",`<br>`        "args": {`<br>`            "cDestIp": "127.0.0.1",`<br>`            "usDestPort": "5011"`<br>`        }`<br>`    }`<br>`}` |

| Associated Plug-Ins | CTF Instructions | Data/Argument Name & Description | Instruction Usage/Application | Examples |
|---|---|---|---|---|
| cfs | ***SendCfsCommandWithPayloadLength*** | Same as ***SendCfsCommand*** with the addition of<br>***payload_length:*** The length in bytes of the command message to send (truncated or padded as needed) | Have CTF modify the length of the payload prior to sending it to the specified target(s). May be used for variable length commands or to intentionally invalidate a command. | `{`<br>`    "instruction":  "SendCfsCommandWithPayloadLength",`<br>`    "data": {`<br>`        "target": "lx_cfs_1",`<br>`        "mid": "TO_CMD_MID",`<br>`        "cc": "TO_ENABLE_OUTPUT",`<br>`        "payload_length": "10"`<br>`    }`<br>`}` |
| cfs | ***SendCfsCommandWithRawPayload*** | ***mid*** : The message ID of the command<br>***cc*** : The command code for the command<br>***hex_buffer*** : A string of hexadecimal characters representing the raw byte value of the command payload<br>***target*** : name of a registered cFS target; optional; default is empty string; if not specified, it applies to all registered cFS targets<br>***header*** : Optional; object where the key is the header field name, and the value is the field value. Fields depend on the CCSDS header implementation. | Have CTF send a command, truncated to the size of a single hexadecimal value, to the specified target(s) | `{`<br>`    "instruction":  "SendCfsCommandWithRawPayload",`<br>`    "data": {`<br>`        "target": "lx_cfs_1",`<br>`        "mid": "DUMMY_IO_CMD_MID",`<br>`        "cc": "DUMMY_IO_RAW_BYTE_CC",`<br>`        "hex_buffer": "0123456789ABCDEF"`<br>`   }`<br>`}` |
| cfs | ***CheckEvent*** | ***target:*** name of a registered cFS target; optional; default is empty string; if not specified, it applies to all registered cFS targets<br>args: a list of objects with the following:<br>***app_name:*** the name of the cFS application that sent the event message<br>***event_id:*** cFS event ID<br>***event_str:*** the event message; optional; default is empty string; if specified, CTF performs a string comparison between the expected and actual strings<br>***is_regex:*** boolean flag to denote a regex match instead of string comparison is to be performed on ***event_str*** ; optional; default is false<br>***event_str_args:*** arguments that will be inserted into ***event_str*** , similar to printf() function; optional; default is empty string | Have CTF verify that one or more cFS events matching the specified parameters have been received from the specified cFS target | `{`<br>`    "instruction":  "CheckEvent",`<br>`    "data": {`<br>`        "target": "lx_cfs_1",`<br>`        "args": [`<br>`           {`<br>`            "app_name": "BEX",`<br>`            "event_id": "13",`<br>`            "event_str": "Processed MODE(%d) Cmd Rcvd",`<br>`            "is_regex": false,`<br>`            "event_str_args": "(1)"`<br>`            }`<br>`        ]`<br>`    }`<br>`}` |

| Associated Plug-Ins | CTF Instructions | Data/Argument Name & Description | Instruction Usage/Application | Examples |
|---|---|---|---|---|
| cfs | ***CheckNoEvent*** | Same as ***CheckEvent*** | Have CTF verify that one or more cFS events matching the specified parameters <u>have not</u> been received from the specified cFS target | `{`<br>`"instruction":  "CheckNoEvent",`<br>`"data": {`<br>`"target": "lx_cfs_1",`<br>`"args": [`<br>`{`<br>`"app_name": "TO",`<br>`"event_id": "3",`<br>`"event_str": "TO - ENABLE_OUTPUT Cmd Rcvd",`<br>`"is_regex": false,`<br>`"event_str_args": ""`<br>`}`<br>`]`<br>`}`<br>`}` |
| cfs | ***CheckTlmValue*** | ***target:*** name of a registered cFS target; optional; default is empty string; if not specified, it applies to all registered cFS targets<br>***mid:*** CCSDS message ID<br>***args:*** array of arguments to be verified; see CTF documentation or example test scripts for additional examples | Have CTF verifies that a telemetry matching the specified parameters <u>has</u> been received from the specified cFS target | `{`<br>`"instruction":  "CheckTlmValue",`<br>`"data": {`<br>`"target": "lx_cfs_1",`<br>`"mid": "TO_HK_TLM_MID",`<br>`"args": [`<br>`{`<br>`"variable": "usCmdErrCnt",`<br>`"compare": "<",`<br>`"value": "1"`<br>`}`<br>`]`<br>`}`<br>`}` |
| cfs | ***CheckTlmPacket*** | ***target:*** name of a registered cFS target; optional; default is empty string; if not specified, it applies to all registered cFS target<br>***mid:*** CCSDS message ID | Have CTF verifies that a telemetry matching the specified MID <u>has</u> been received from the specified cFS target | `{`<br>`"instruction":  "CheckTlmPacket",`<br>`"data": {`<br>`"target": "lx_cfs_1",`<br>`"mid": "TO_HK_TLM_MID"`<br>`}`<br>`}` |
| cfs | ***CheckNoTlmPacket*** | ***target:*** name of a registered cFS target; optional; default is empty string; if not specified, it applies to all registered cFS target<br>***mid:*** CCSDS message ID | Have CTF verifies that a telemetry matching the specified MID <u>has not</u> been received from the specified cFS target | `{`<br>`"instruction":  "CheckNoTlmPacket",`<br>`"data": {`<br>`"target": "lx_cfs_1",`<br>`"mid": "TO_HK_TLM_MID"`<br>`}`<br>`}` |
| cfs | ***CheckTlmContinuous*** | Same as ***CheckTlmValue*** , with the addition of<br>***verification_id:*** a unique ID to identify this verification <u>within</u> a test case | Similar to ***CheckTlmValue*** , except the verification is done every time the telemetry is received, until the test case ends, or the check is removed by ***RemoveCheckTlmContinuous*** | `{`<br>`"instruction":  "CheckTlmContinuous",`<br>`"data": {`<br>`"target": "lx_cfs_1",`<br>`"verification_id": "TO_no_errors",`<br>`"mid": "TO_HK_TLM_MID",`<br>`"args": [`<br>`{`<br>`"variable": "usCmdErrCnt",`<br>`"compare": "<",`<br>`"value": "1"`<br>`}`<br>`]`<br>`}`<br>`}` |

| Associated Plug-Ins | CTF Instructions | Data/Argument Name & Description | Instruction Usage/Application | Examples |
|---|---|---|---|---|
| cfs | ***RemoveCheckTlmContinuous*** | ***target:*** name of a registered cFS target; optional; default is empty string; if not specified, it applies to all registered cFS targets<br>***verification_id:*** the verification_id value of an existing ***CheckTlmContinuous*** | Have CTF stop performing continuous telemetry check for ***CheckTlmContinuous*** matching the ***verification_id*** parameter | {<br>"instruction": "RemoveCheckTlmContinuous",<br>"data": {<br>"verification_id": "TO_no_errors"<br>}<br>} |
| cfs | ***ArchiveCfsFiles*** | ***target:*** name of a registered cFS target; optional; default is empty string; if not specified; it applies to all registered cFS targets<br>***source_path:*** a directory path to copy from, either absolute or relative to the location of CTF | Have CTF copy files from the specified directory into the test run's log directory | {<br>"instruction": "ArchiveCfsFiles",<br>"data": {<br>"target": "lx_cfs_1",<br>"source_path": "../../build/exe/lx_cfs_1/cf/"<br>}<br>} |
| cfs | ***ShutdownCfs*** | ***target:*** name of a registered cFS target; optional; default is empty string; if not specified; it applies to all registered cFS targets | Have CTF shut down cFS target(s) within the test script. CTF will automatically shutdown cFS targets upon test completion. | {<br>"instruction": "ShutdownCfs",<br>"data": {<br>"target": "lx_cfs_1"<br>}<br>} |
| ssh | ***SSH_RegisterTarget*** | ***name:*** an arbitrary-but-unique name to identify an SSH target | Register an SSH target with the specified name so CTF can manage its external interfaces. CTF can manage multiple SSH targets. | {<br>"instruction": "SSH_RegisterTarget",<br>"data": {<br>"name": "ssh_tgt_1"<br>}<br>} |
| ssh | ***SSH_InitSSH*** | ***host:*** network hostname or IP to connect to<br>***name:*** a registered SSH target; optional<br>***user:*** user name for the connection; do not specified if already included in ***host*** ; optional<br>***port:*** port number for the connection; optional<br>***gateway:*** SSH gateway command to proxy the connection to ***host*** ; optional<br>***ssh_config_path:*** path to an ssh config file that contains host definitions or additional parameters; optional; default is "~/.ssh/config"<br>***args:*** additional SSH connection options, as needed | Have CTF establish the SSH connection with a target host | {<br>"instruction": "SSH_InitSSH",<br>"data": {<br>"name": "ssh_tgt_1",<br>"host": "123.123.123.1",<br>"user": "lander_demo",<br>"port": 22,<br>"gateway": "ssh -W %h:%p myproxy",<br>"ssh_config_path": "./ssh/config"<br>}<br>} |
| ssh | ***SSH_RunRemoteCommand*** | ***name:*** a registered SSH target<br>***command:*** the shell command to be executed; can contain multiple commands separated with semicolon<br>***cwd*** : current working directory in which to execute command; optional<br>***prefix*** : an additional command to execute before command whose result is ignored; optional | Have CTF execute a shell command on the SSH target | {<br>"instruction": "SSH_RunRemoteCommand",<br>"data": {<br>"name": "ssh_tgt_1",<br>"command": "cd lander_fsw_ctf/;rm -rf build; make; make install;"<br>}<br>} |

| Associated Plug-Ins | CTF Instructions | Data/Argument Name & Description | Instruction Usage/Application | Examples |
|---|---|---|---|---|
| ssh | ***SSH_RunLocalCommand*** | ***name:*** a registered SSH target; optional<br>***command:*** the shell command to be executed; can contain multiple commands separated by semicolons | Have CTF execute a shell command on the local host | `{`<br>`    "instruction":  "SSH_RunLocalCommand",`<br>`    "data": {`<br>`        "name": "ssh_tgt_1",`<br>`        "command": "cd lander_fsw_ctf/;rm -rf build; make; make install;"`<br>`    }`<br>`}` |
| ssh | ***SSH_CheckOutput*** | ***name:*** a registered SSH target<br>***output_contains:*** a substring that must be contained in the output<br>***output_does_not_contain:*** a substring that should not be contained in the output; optional<br>***exit_code:*** the expected exit code after the command is executed | Have CTF verify the output of the most recently executed SSH command | `{`<br>`    "instruction":  "SSH_CheckOutput",`<br>`    "data": {`<br>`        "name": "ssh_tgt_1",`<br>`        "output_contains": "Built target mission-install",`<br>`        "output_does_not_contain": "Error",`<br>`        "exit_code": 0`<br>`    }`<br>`}` |
| ssh | ***SSH_PutFile*** | ***name:*** a registered SSH target<br>***local_path:*** path to the local file/directory as source path<br>***remote_path:*** path to the remote file/directory as destination path<br>***args*** : parameters for the transfer<br>***args.delete:*** a boolean option for rsync command with --delete argument; if set to true; rsync will remove remote files that no longer exists locally; optional; default is false<br>***args.exclude:*** a string of option for rsync command with --exclude argument; optional; default is an empty string | Have CTF copy files from the specified local directory into the specified remote directory | `{`<br>`    "instruction":  "SSH_PutFile",`<br>`    "data": {`<br>`        "name": "ssh_tgt_1",`<br>`        "local_path": "./cfs",`<br>`        "remote_path": "/tmp/workspace/cfs",`<br>`        "args": {`<br>`            "delete": true,`<br>`            "exclude": "*.git"`<br>`        }`<br>`    }`<br>`}` |
| ssh | ***SSH_GetFile*** | ***name:*** a registered SSH target<br>***remote_path:*** path to the remote file/directory as source path<br>***local_path:*** path to the local file/directory as destination path<br>***args:*** parameters for the transfer<br>***args.delete:*** a boolean option for rsync command with --delete argument; if set to true; rsync will remove remote files that no longer exists locally; optional; default is false<br>***args.exclude:*** a string of option for rsync command with --exclude argument; optional; default is an empty string | Have CTF copy files from the specified remote directory into the specified local directory | `{`<br>`    "instruction":  "SSH_GetFile",`<br>`    "data": {`<br>`        "name": "ssh_tgt_1",`<br>`        "local_path": "./results.txt",`<br>`        "remote_path": "./data/output.dat"`<br>`    }`<br>`}` |

| Associated Plug-Ins | CTF Instructions | Data/Argument Name & Description | Instruction Usage/Application | Examples |
|---|---|---|---|---|
| ssh | ***SSH_GetFTP*** | ***name:*** a registered SSH target<br>***host:*** hostname or address of the FTP server<br>***remote_path:*** remote path to the file/directory is to be downloaded from<br>***local_path:*** local path to where the file/directory is to be downloaded to | Have CTFdownload file/directory from a remote directory via FTP | `{`<br>`    "instruction":  "SSH_GetFTP",`<br>`    "data": {`<br>`        "name": "ssh_tgt_1",`<br>`        "host": "ftphost",`<br>`        "local_path": "./results.txt",`<br>`        "remote_path": "./data/output.dat"`<br>`    }`<br>`}` |
| ssh | ***SSH_PutFTP*** | ***name:*** a registered SSH target<br>***host:*** hostname or address of the FTP server<br>***local_path:*** local path to where the file/directory is to be uploaded from<br>***remote_path:*** remote path to the file/directory on the FTP server where local file/directory is to be uploaded to | Have CTF upload file/directory to a remote directory via FTP | `{`<br>`    "instruction":  "SSH_PutFTP",`<br>`    "data": {`<br>`        "name": "ssh_tgt_1",`<br>`        "host": "ftphost",`<br>`        "local_path": "./results.txt",`<br>`        "remote_path": "./data/output.dat"`<br>`    }`<br>`}` |
| userio | ***WaitForUserInput*** | ***prompt:*** a string to be displayed to prompt for user input; optional; default is an empty string | Have CTF pause the test execution until it receives user input to either resume or stop | `{`<br>`    "instruction":  "WaitForUserInput",`<br>`    "data": {`<br>`        "prompt": "Enter Y or N:",`<br>`    }`<br>`}` |
| variable | ***SetUserVariable*** | ***variable_name:*** user-defined variable name<br>***operator*** : assignment or math operator to apply to variable (e.g.: =, +, -, *, /)<br>***value:*** value of primitive data type (e.g., integer, float or string) | Have CTF create a variable and assign a value to it; commonly use for loop counter | `{`<br>`    "instruction":  "SetUserVariable",`<br>`    "data": {`<br>`        "variable_name": "my_var_1",`<br>`        "operator": "=",`<br>`        "value": 10`<br>`    }`<br>`}` |
| variable | ***SetUserVariableFromTlm*** | ***variable_name:*** user-defined variable name<br>***mid:*** CCSDS message ID<br>***tlm_variable:*** name of the telemetry item as defined in the json definition for that telemetry message | Have CTF create a variable and assign a value of a telemetry item to it | `{`<br>`    "instruction":  "SetUserVariableFromTlm",`<br>`    "data": {`<br>`        "variable_name": "my_var_2",`<br>`        "mid": "TO_HL_TLM_MID",`<br>`        "tlm_variable": "usCmdCnt"`<br>`    }`<br>`}` |
| variable | ***SetUserVariableFromTlmHeader*** | ***variable_name*** : user-defined variable name<br>***mid*** : CCSDS message ID<br>***header_variable*** : name of the telemetry item as defined in the json definition for CCSDS headers | Have CTF create a variable and assign a value of a telemetry header field to it | `{`<br>`    "instruction":  "SetUserVariableFromTlmHeader",`<br>`    "data": {`<br>`        "variable_name": "my_var_2",`<br>`        "mid": "TO_HL_TLM_MID",`<br>`        "header_variable": "pheader.length"`<br>`    }`<br>`}` |
| variable | ***CheckUserVariable*** | ***variable_name:*** user-defined variable name<br>***operator:*** comparison operator to apply to variable<br>***value:*** value of primitive data type (e.g., integer, float or string) | Have CTF compare the value of an existing user-defined variable to the specified value | `{`<br>`    "instruction":  "CheckUserVariable",`<br>`    "data": {`<br>`        "variable_name": "my_var_1",`<br>`        "operator": "==",`<br>`        "value": 4`<br>`    }`<br>`}` |

| Associated Plug-Ins | CTF Instructions | Data/Argument Name & Description | Instruction Usage/Application | Examples |
|---|---|---|---|---|
| control_flow | ***BeginLoop*** | ***label:*** user-defined label; must be unique within the scope of the test case<br>***conditions:*** a list of comparison conditions<br>***condition.variable:*** name of an existing user-defined variable<br>***condition.compare:*** comparison operator<br>***condition.value:*** value of primitive data type | Have CTF create a loop entry point and associate it with the specified looping label; commonly use to loop thru a set of test instructions | {<br>"instruction": "BeginLoop",<br>"data": {<br>"label": "loop_1",<br>"conditions": [<br>{ "variable": "my_var_1",<br>"compare": "<",<br>"value": 4<br>},<br>{ "variable": "my_var_2",<br>"compare": ">",<br>"value": 1<br>}<br>]<br>}<br>} |
| control_flow | ***EndLoop*** | ***label:*** user-defined label; must match an existing label from ***BeginLoop*** instruction | Have CTF create a loop exit point and associate it with the specified looping label | {<br>"instruction": "EndLoop",<br>"data": {<br>"label": "loop_1"<br>}<br>} |
| control_flow | ***IfCondition*** | ***label*** : user-defined label; must be unique within the scope of the test case<br>***conditions*** : a list of comparison conditions<br>***condition.variable*** : name of an existing user-defined variable<br>***condition.compare*** : comparison operator<br>***condition.value*** : value of primitive data type | Have CTF create a conditional branch entry point and associate it with the specified label; commonly use to execute or skip a set of test instructions based on comparision results. | {<br>"instruction": "IfCondition",<br>"data": {<br>"label": "If_Label_1",<br>"conditions": [<br>{ "variable": "my_var_1",<br>"compare": "<",<br>"value": 4<br>},<br>{ "variable": "my_var_2",<br>"compare": ">",<br>"value": 1<br>}<br>]<br>}<br>} |
| control_flow | ***ElseCondition*** | ***label*** : user-defined label; must match an existing label from ***IfCondition*** instruction | Have CTF create an entry point for else conditional branch block and associate it with the specified label. If the condition of IfCondition instruction is False, the control flow skips the 'if' branch block, and executes the 'else' branch block | {<br>"instruction": "ElseCondition",<br>"data": {<br>"label": "If_Label_1"<br>}<br>} |
| control_flow | ***EndCondition*** | ***label*** : user-defined label; must match an existing label from ***IfCondition*** instruction | Have CTF create a ifcondition exit point and associate it with the specified label | {<br>"instruction": "EndCondition",<br>"data": {<br>"label": "If_Label_1"<br>}<br>} |
| validation | ***CopyFiles*** | ***source:*** path of the file / folder to be copied from on host machine<br>***destination:*** path of the file / folder to be copied to on host machine | Copy a file or folder on the host file system. If the destination exists, it will be overridden. | {<br>"instruction": "CopyFiles",<br>"data": {<br>"source": "./testArtifacts1",<br>"destination": "./testArtifacts2"<br>}<br>} |

| Associated Plug-Ins | CTF Instructions | Data/Argument Name & Description | Instruction Usage/Application | Examples |
|---|---|---|---|---|
| validation | ***DeleteFiles*** | ***path:*** path of the file / folder to be deleted on host machine. | Delete a file or folder on the host file system. | `{`<br>`    "instruction":  "DeleteFiles",`<br>`    "data": {`<br>`        "path": "./testArtifacts1"`<br>`    }`<br>`}` |
| validation | ***SearchStr*** | ***file:*** path of the text file<br>***search_str:*** text string to search | Search a text file for a given text string. | `{`<br>`    "instruction":  "SearchStr",`<br>`    "data": {`<br>`        "file": "/testArtifacts/event_log.txt",`<br>`        "search_str": "computer reboot"`<br>`    }`<br>`}` |
| validation | ***SaveFileAsText*** | ***input_file*** : path of the binary event log file<br>***output_file*** : path of the output text file.<br>***file_type*** : currently only supports 'EVS' file type<br>***target*** : cfs target, optional | Save the cFE event log file (binary file created via the CFE_EVS_WRITE_LOG_DATA_FILE_CC command) to a human-readable text file | `{`<br>`    "instruction":  "SaveFileAsText",`<br>`    "data": {`<br>`        "input_file": "/dev/shm/osal:RAM/event.bin",`<br>`        "output_file": "./testArtifacts/event_log.txt",`<br>`        "file_type": "EVS",`<br>`        "target": "cfs"`<br>`    }`<br>`}` |