Last edited by **Reto Da Forno** 1 month ago

XmlConfig

FlockLab XML Test Configuration File Help

The FlockLab XML test configuration file defines everything needed for a single FlockLab test: when a test should start and stop, which platform and operating system are used, which services are used and how the test results are going to be retrieved.

- The XML is validated against an XML schema definition.
- We provide a sample configuration file (including an image for the `dpp` (CC430 sensor node))
- Test configuration files can be validated (without being actually registered in FlockLab).
- Note that the ordering of the XML tags matters within a configuration block matters.

👉 If you have an old XML config of FlockLab 1, you can adjust the config according to this porting guide.

Structure of the test configuration

A test configuration consists of several blocks:

- A block with the general configuration of the test: when a test starts/ends, which targets are used and so on.
- Blocks with the configurations for all targets: which image is used, which ID an image should have and the used voltage
- Blocks for each service used in the test. These blocks are only needed for services which are actually needed. All configuration elements have to be embedded into the outermost element:

```
<?xml version="1.0" encoding="UTF-8"?>
<testConf xmlns="http://www.flocklab.ethz.ch" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.fl
  ...
</testConf>
```

Next, place the needed configuration elements into this structure:

1. `<generalConf>` : 1 element has to be specified.
2. `<targetConf>` : 1 or more elements have to be specified.
3. `<serialConf>` : 0 or more elements can be specified. Do only specify if needed.
4. `<gpioTracingConf>` : 0 or more elements can be specified. Do only specify if needed.
5. `<gpioActuationConf>` : 0 or more elements can be specified. Do only specify if needed.
6. `<powerProfilingConf>` : 0 or more elements can be specified. Do only specify if needed.
7. `<embeddedImageConf>` : 0 or more elements can be specified. Do only specify if needed.
8. `<debugConf>` : 0 or more elements can be specified. Do only specify if needed.

General Configuration

The element `<generalConf>` defines the general test configuration.

Element	***Description***	***Format***
`name` *	A short, meaningful name for the test.	5 - 45 characters
`description`	More comprehensive description of the test.	max. 512 characters
`custom`	A custom tag, can contain arbitrary user defined data. Note: tags `<...>` are not allowed within this element.	max. 4096 characters
`schedule` *	Defines the test start time and duration.	

Element	*Description*	*Format*
`emailResults`	If set, the test results are emailed to the user by the end of the test. The email address is taken from the user account settings.	possible values: `yes` , `no` . Default: `no`
`abortOnError`	Whether the test should be aborted if an error is detected during the test setup. By default FlockLab will try to run the test even if some services or observers fail to start the test. If this option is enabled, the test will be aborted on *all* observers if an error is detected during test setup on any of the used observers.	possible values: `yes` , `no` . Default: `no`

* mandatory elements

`<schedule>`

Element	*Description*	*Format*
`start`	Start time in UTC of the test. Check the schedule for free slots. If no start time is provided, the test will run as soon as possible.	`YYYY-mm-ddTHH:MM:SSZ` , e.g. 2011-08-20T16:00:00Z
`duration` *	Duration of the test in seconds.	min. is 20 seconds, max. is limited by the users quota [1]

* mandatory elements
[1] During office hours (0700 - 1700 GMT), the max. scheduled test duration per user is limited to 30min.

Target Configuration

All targets used in a test need to be defined within the `<targetConf>` element.

Element	*Description*	*Format*
`obsIds` *	One or more ID of the observer(s) to be configured. If you want to use all available observers, you can specify `ALL` instead of listing all observer IDs. -> How to use obsId in the code	Observer ID(s) separated by spaces. Consult the nodelist page for a list of usable observers and targets attached to the observers.
`targetIds`	Optional user-defined ID(s) for the target(s). The node ID(s) of the target(s) will be set to these values. For each observer ID in `<obsIds>` , a corresponding target ID is required in `<targetIds>` . -> How to use targetId in the code	16bit unsigned integer(s) separated by spaces.
`voltage`	Voltage in volt with which the target is to be powered.	1.8 - 3.3 in 0.1 steps. If element is not specified, 3.3 is assumed.
`dbImageId` *	Image ID from the FlockLab database to be used on this target(s). Consult the images management site for a list of available images. Note that an image has to be uploaded to the images management site before it can be referenced here. Either `<dbImageId>` or `<embeddedImageId>` has to be specified.	
`embeddedImageId` *	Image ID referencing an `<embeddedImageConf>` element in the XML configuration containing an image binary. Either `<dbImageId>` or `<embeddedImageId>` has to be specified.	Any combination of characters, numbers and underline

* mandatory elements

Embedded Image Configuration

`<embeddedImageConf>` . Target images can be defined in two ways for a test: either by referencing an image which is already in the FlockLab database or by adding the image directly to the XML configuration file. The latter case is done with `<embeddedImageConf>` elements.

Element	*Description*	*Format*
`embeddedImageId` *	Image ID referenced by `<targetConf>` element.	any combination of characters, numbers and underline

Element	***Description***	***Format***
`name` *	A short, meaningful name for the image.	5 - 45 characters
`description`	More comprehensive description of the image.	max. 512 characters
`platform` *	Target platform for which the image is intended.	Possible values: `tmote` , `dpp` , `dpp2lora` , `nrf5`
`core`	Specify the core/MCU, only of interest for platforms with more than one MCU.	for the DPP: 0 = CC430, 1 = BOLT, 2 = MSP432
`data` *	The image itself. You can use this script to embed an image into the XML file.	base64 encoded string of exe/elf or Intel hex file

* mandatory elements

Serial Service Configuration

To use the serial service, define a `<serialConf>` block with the following configuration elements:

Element	***Description***	***Format***
`obsIds` *	One or more observer IDs for which this service should be used. Note that all IDs used here have to have a target configuration as well.	ID(s) of the observer(s) to use
`port`	Which port to use for serial communication. *Note 1*: In order for port `swo` to work, the SWD debug pins (SWDIO, SWDCLK) must not be reconfigured as output pins. *Note 2*: for port `serial` the `baudrate` element is requried, for port `swo` the `cpuSpeed` element is required!	Possible values: `usb` , `serial` , `swo` . For the target Tmote, the default is `usb` , for all other platforms `serial` is the default.
`baudrate`	Baud rate to use when reading. Note that baud rates higher than 460800 may not work properly on all targets, i.e. characters can get lost and bit errors are more likely.	Possible values: 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600, 1000000, 1152000, 1500000, 2000000, 2500000, 3000000, 3500000, 4000000. Default is 115200
`cpuSpeed`	Defines the target CPU speed (required if port `swo` is selected).	Frequency in Hertz. Default is 80000000.
`remoteIp`	IP address from which serial data will be sent to node. If not specified, the serial proxy service will not be used. If left blank, the IP of the machine that registered the test will be used. Notes: - If your machine only has an IPv6 address, then the XML validation will fail. - Timestamping will be more accurate if the proxy service is not used (no `<remoteIp>` element in the XML config).	IPv4 address

* mandatory elements

How to connect to the serial service.

GPIO Tracing Service Configuration

To use the GPIO tracing service, define a `<gpioTracingConf>` block with the following configuration elements:

Element	***Description***	***Format***
`obsIds` *	One or more observer IDs for which this service should be used. Note that all IDs used here have to have a target configuration as well.	ID(s) of the observer(s) to use
`pins` *	Defines which pin to trace. See GPIO list for pin assignments. Important: make sure the pins you want to trace are never in a floating / undefined state!	possible values: `INT1` , `INT2` , `LED1` , `LED2` , `LED3`
`offset`	Postpone the sampling start. Offset relative to the start of the test, in seconds. Default value is 1.	unsigned integer

* mandatory elements

GPIO Actuation Service Configuration

Remark: the default level of both actuation pins (`SIG1` , `SIG2`) is low. If the actuation service is not used, the actuation pins will remain in low state during the whole test. In addition to the `SIGx` pins, the target reset (`nRST`) can also be actuated.

To use the GPIO actuation service, define a `<gpioActuationConf>` block with the following configuration elements:

Element	***Description***	***Format***
`obsIds` *	One or more observer IDs for which this service should be used. Note that all IDs used here have to have a target configuration as well.	ID(s) of the observer(s) to use
`pinConf` *	One or more blocks containing configuration for one pin.	(see below)

* mandatory elements

`<pinConf>`

Element	***Description***	***Format***
`pin` *	Subelement of `<pinConf>` . Defines which pin to set/clear/toggle (event). See GPIO list for pin assignments.	possible values: `SIG1` , `SIG2` , `nRST`
`level` *	Subelement of `<pinConf>` . Defines to which level the pin is to be actuated. If a period is specified, level `toggle` is implied.	possible values: `low` , `high` , `toggle`
`offset` *	Subelement of `<pinConf>` . Defines an event relative to the start time of the whole test (offset in seconds).	float value between 0.0 and 4000.0
`period`	Subelement of `<pinConf>` . If set, the event will be periodic with the chosen period (in seconds). Within one period, the pin will be toggled twice.	float value between 0.001 and 1000.0
`count`	Subelement of `<pinConf>` . Defines the number of cycles for periodic events. One cycle lasts for one period and consists of two pin toggling events.	int value between 1 and 4000

* mandatory elements

Power Profiling Service Configuration

To use the power profiling service, define a `<powerProfilingConf>` block with the following configuration elements:

Element	***Description***	***Format***
`obsIds` *	One or more observer IDs for which this service should be used. Note that all IDs used here have to have a target configuration as well.	ID(s) of the observer(s) to use
`offset` *	Defines the start of the power measurement relative to the test start (offset in seconds).	
`duration`	Duration in seconds of the power measurement.	
`samplingRate`	Sampling rate [Hz] for the power measurement.	possible values: 1, 10, 100, 1000, 2000, 4000, 8000, 16000, 32000, 64000. default: 1000
`fileFormat`	File format. The csv format is more convenient (text), but results in large files. If you want to parse the data manually, you can use the rld format (binary). Parsing tools are available here.	possible values: csv, rld. default: csv
`aggregate`	Targeted sampling rate after aggregation, must be smaller than the sampling rate. If set, the results will be downsampled (averaged) to the specified rate in Hertz. **Note:** This option can only be used with file format 'csv'.	possible values: 0, 1, 10, 100, 1000, 2000, 4000, 8000, 16000, 32000. default: 0 (no aggregation)

* mandatory elements

Debug Configuration

If you want to use the debug service, define a `<debugConf>` block with the following configuration elements:

Element	***Description***	***Format***
`obsIds` *	One or more observer IDs for which this service should be used.	ID(s) of the observer(s) to use
`gdbPort`	If specified, a GDB server will be started on the observer and listening for incoming connections. **Note:** The debug server is only started 10 seconds after the scheduled test start time!	possible values: 2331 **Remark:** SWV (SWO) will be available on port 2332
`cpuSpeed`	Specifies the target CPU speed. Only required (and allowed!) if `dataTraceConf` is used.	Frequency in Hertz. Default is 80000000.
`dataTraceConf`	Config for data trace of one global variable (up to 4 `dataTraceConf` blocks are supported).	(see below)

* mandatory elements

Note: `gdbPort` and `dataTraceConf` are exclusive options and cannot be used simultaneously on the same observer.

`<dataTraceConf>`

Trace the value (32-bit) of a global variable with the ARM Cortex M4 CoreSight DWT feature.

Element	***Description***	***Format***
`variable` *	Name or address of the global variable to trace. **Note**: The address (or the address of the variable) must be properly aligned (i.e. the address must be an integer multiple of `size`) otherwise the resulting trace is unpredictable!	string (in case the address is provided, it must have the prefix `0x`)
`mode`	Access mode to trace. Example: `R` (read) means only read accesses to the variable will be logged. If `PC` is appended (`R PC`), the program counter will also be recorded.	possible values: `R` (read), `W` (write), `RW` (read + write), `PC` (program counter) default: `W`
`size`	Size of the variable to be traced, in bytes. Note that the address has to be aligned to the respective size.	integer (powers of two only: 1, 2, 4, 8, ...), default: 4

* mandatory elements

Notes:

- The data trace feature only works with ARM Cortex M4 targets, i.e. `DPP2LoRa` and `nRF5` . The target image in `<targetConf>` has to be provided in the ELF file format.
- Correct CPU frequency needs to be set with the `cpuSpeed` config element (see `debugConf` above).
- SWO pin needs to be configured in SWO mode.
- For the target DPP2LoRa: If you want to use the pins SWDCLK and SWDIO for GPIO tracing, you must add a short startup delay in the target code before reconfiguring these pins as outputs. Otherwise the data trace service will not be able to configure the MCU properly.