Shell front-end

1. API

A) Execute shell commands from preset.

i) Use IOCTL_SHELL_FRONTEND_LOAD_PRESET ioctl number.

ii) Pointer to struct shell_frontend_load_preset_t should passed in first ioctl data parameter (3rd argument).

```
struct shell_frontend_load_preset_t {
        struct dev_desc_t *shell_preset_pdev; // - pointer to preset manager module
        uint8_t num_of_preset;                // - number of preset to load
};
```

iii) This call should **never** be called from shell commands.
to load preset from shell command use shell_frontend_load_preset_from_shell_cmd()

iv) The call is non-blocking – so return from this function does not mean that all preset was already loaded.

B) Execute shell commands from buffer.

i) Use IOCTL_SHELL_FRONTEND_RUN_BATCH ioctl number.

ii) Pointer to struct shell_frontend_load_preset_t should passed in first ioctl data parameter (3rd argument).

```
struct shell_frontend_batch_t {
        uint8_t *batch_buffer;      // - buffer of command
        size_t batch_buffer_size;  // - size of buffer
};
```

iii) This call should **never** be called from shell commands.

iv) The call is non-blocking – so return from this function does not mean that all preset was already loaded.

v) Example:

```
static uint8_t cmd_buf[] =
            "cmd1 \r\n"
            "cmd2 \r\n";
            "cmd3 \r\n";
               ...
               ...
               ...
  struct shell_frontend_batch_t shell_frontend_batch;

  shell_frontend_batch.batch_buffer = cmd_buf;
  shell_frontend_batch.batch_buffer_size = sizeof(cmd_buf) - 1;
  ret_val = DEV_IOCTL_1_PARAMS(shell_frontend_dev ,
                    IOCTL_SHELL_FRONTEND_RUN_BATCH, &shell_frontend_batch);
```