1. MIDI-MT is High Level Application Driver for USB MIDI Control Surface

1.0.1. Dependencies

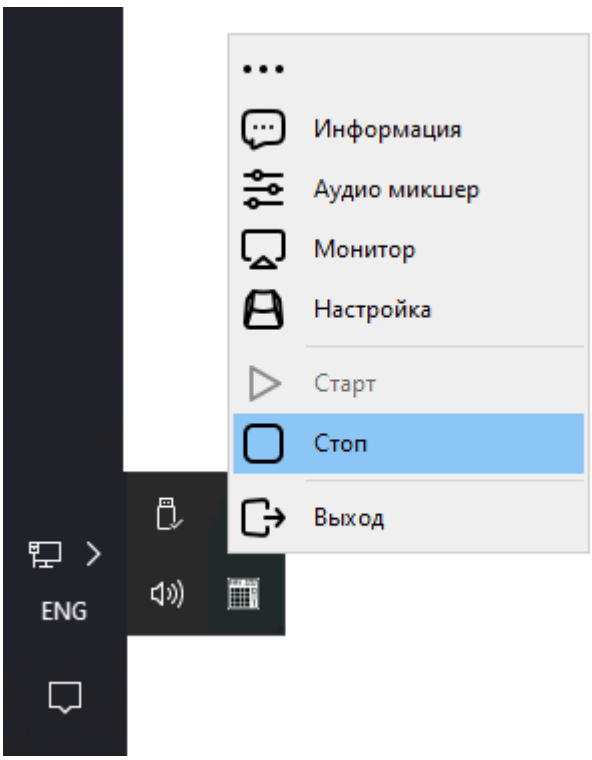
The application uses MIDI drivers from Tobias Erichsen, version **1.3.0.43** dated 2019-12-02. To install the drivers visit the site and install loopMIDI or rtpMIDI. These distributions include the drivers necessary for operation.

If for some reason the site tobias-erichsen.de is not available, loopMIDI can be downloaded from this repository.

The loopMIDI application itself is useful, allowing you to experiment with connecting and switching MIDI equipment. Installation shouldn't be difficult, just download the **MIDIMT_XX.msi** installation package from latest available release and install in a standard way.

1.0.1. Launch

To quickly configure your EasyControl controller, you can use the config file for the EasyControl Setup from the manufacturer. The settings file for the application **MIDIMT** is in Json format and its settings must match the settings of the controller. Description of settings in Json format are described in **Settings**.
After starting the application, you need to start the MIDI translator itself. Find the **MIDIMT** application icon in the tray, open the menu and select **Start**. A dialog will open in which you can change the basic settings when starting the translator.

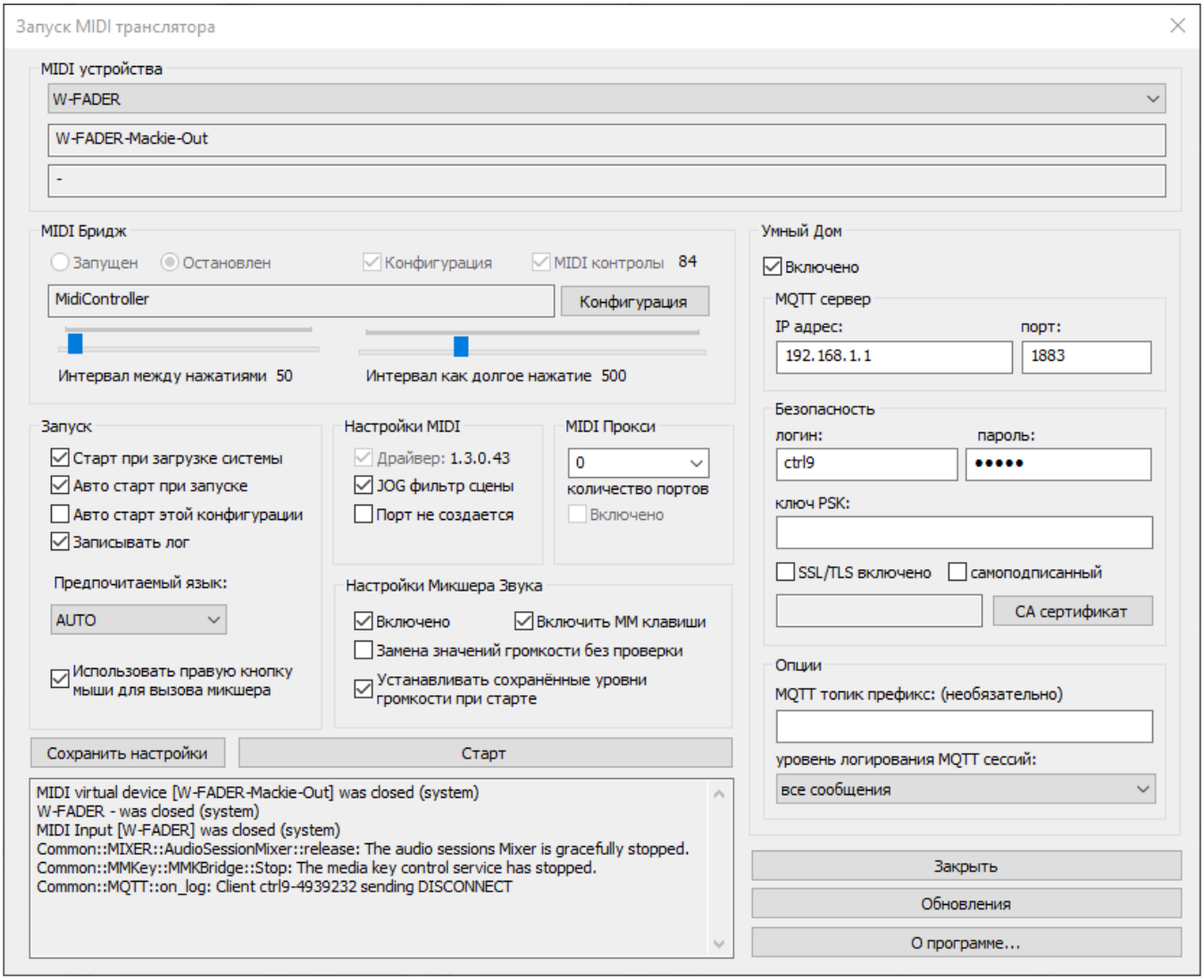


1.1. MIDI port naming convention

You can select the input MIDI port from the drop-down list, it must be a MIDI controller port. The MIDI output port on which MIDI-Mackie notations are generated has a name composed of the name of the input port and the suffix **-Mackie-Out**. The proxy MIDI port name is formed in the same way, the suffix for it will be **-Proxy-Out**. For example:

```
// MIDI controller
MIDI input port: W-FADER
// Premiere Pro (MIDI Mackie) port
MIDI output port: W-FADER-Mackie-Out
// Any application for parallel control from a controller
Proxy MIDI port: W-FADER-Proxy-Out
```

The exact names of the ports can be obtained using the menu item **Info** while running **MIDI-MT**.



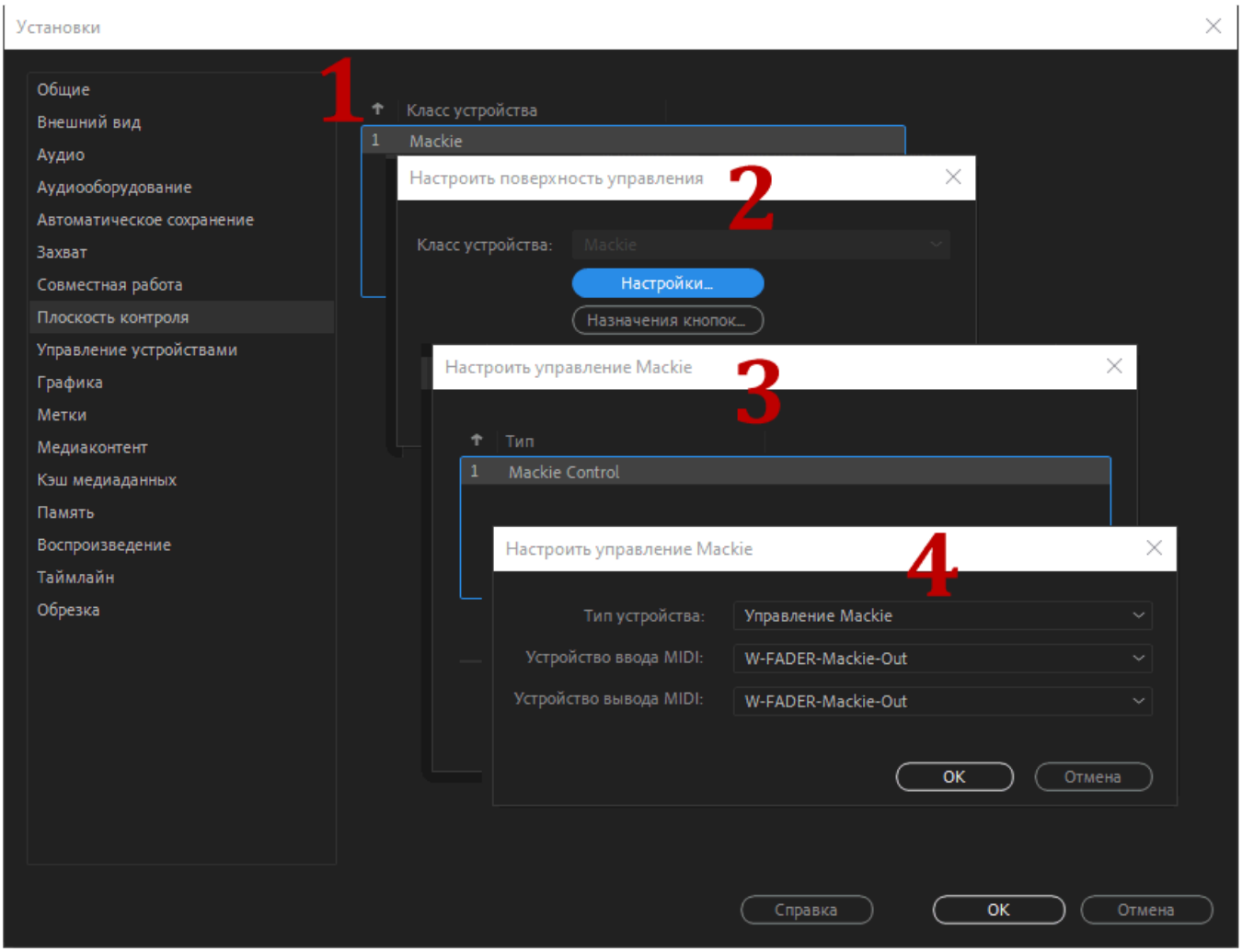
1.2. Behavior

- the **auto start** item is checked, the dialog will not wait for the **Start** button to be pressed, and at the moment of opening it will start the translator.
- the **start at boot** item is checked, at the moment the application automatically starts when the operating system boots, the **MIDI-MT** application will be loaded. If the **auto start** item is active, the translator will start automatically.
- the **write log** item is responsible for recording the events in the file, for possible further analysis of the failure/disconnection/connection of MIDI equipment.
- the **Monitor** item shows whether the output of MIDI commands is connected for debugging and equipment settings. It is worth noting that this item is turned on and off automatically from the **Monitor** dialog called from the menu. You should not change its state, the status display is more informational and shows whether the MIDI hardware debug mode is currently connected.
- the **Proxy** item is responsible for creating output proxy MIDI ports. If a value other than 0 is selected, the specified number of MIDI output proxy ports are created into which all MIDI traffic coming from your controller is broadcast. This can be used to connect other software that can also be controlled from your EasyControl MIDI controller. - if **Port not created** is checked, **MIDI-MT** will not create virtual MIDI ports. You need to create the port yourself in the **loopMIDI** manager according to the port naming convention. This can be useful if for some reason you want to keep the virtual MIDI port working, even if the **MIDI-MT** application is not loaded.
- **MIDI port** and **MIDI connection** items show the status of the MIDI ports.
- The **Interval between presses** adjusts the "bounce of contacts", preventing false positives. The time is set in milliseconds, the default value is 50.
- The **Interval as long press** knob adjusts the response time of the button treated as a "long press". Due to this parameter, it is possible to use the button to control two parameters at once. For example, the default settings of the buttons below the faders include a "short press" of the **Solo** mode, and a "long press" of the **Mute** mode on the selected channel. The time is set in milliseconds, the default value is 500.

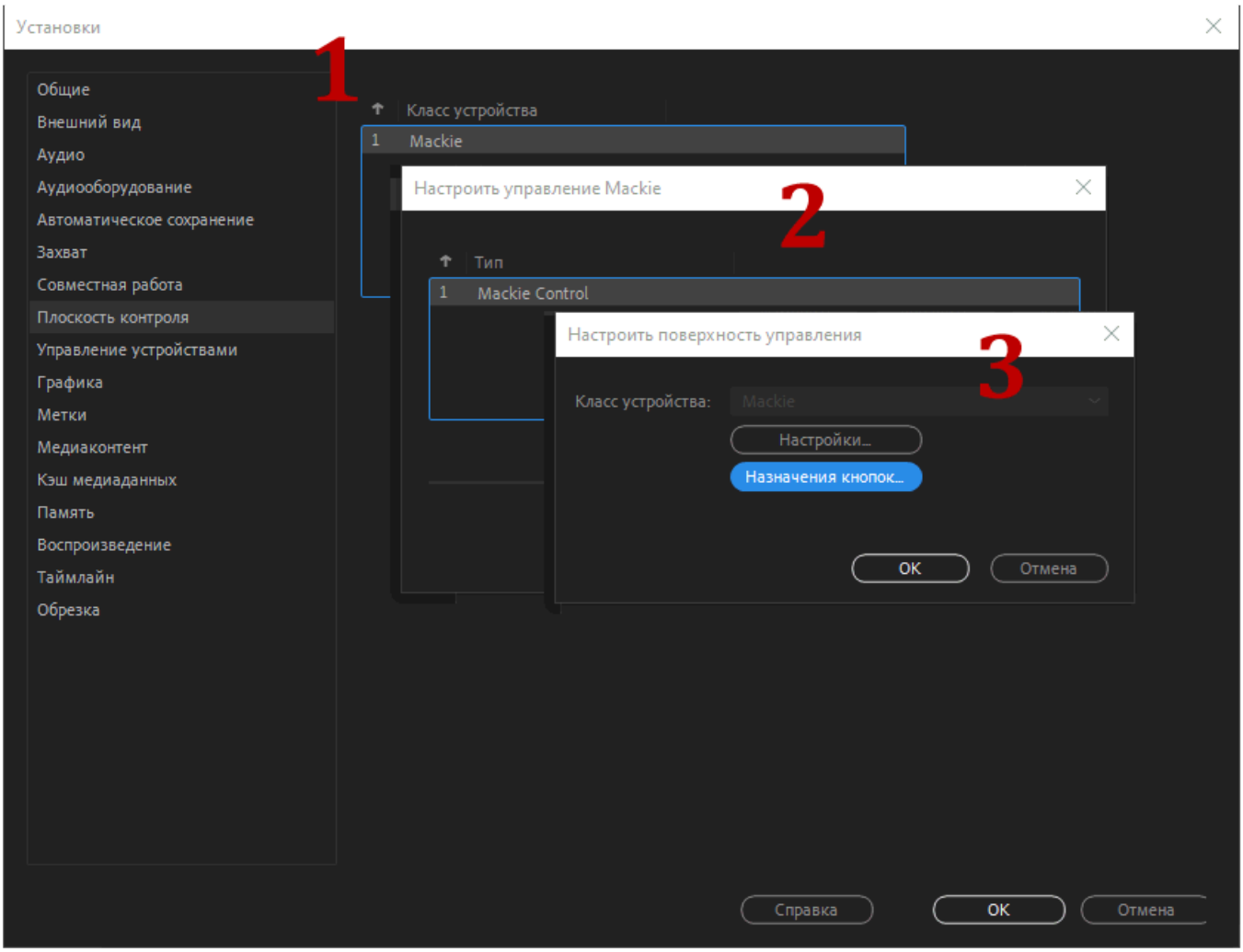
1.3. Operations

If the **auto start** item is not activated, you must press the **Start** button to start the translator. The changes made can be saved in the configuration file using the **Save Settings** button. In the future, these settings will be applied automatically. To exit this dialog, click the **Close** button. The **About** and **Updates** buttons are intuitive and self-explanatory. Setting up **Premier PRO** comes down to selecting a MIDI controller from the **Control Plane** submenu in the **Settings** menu.

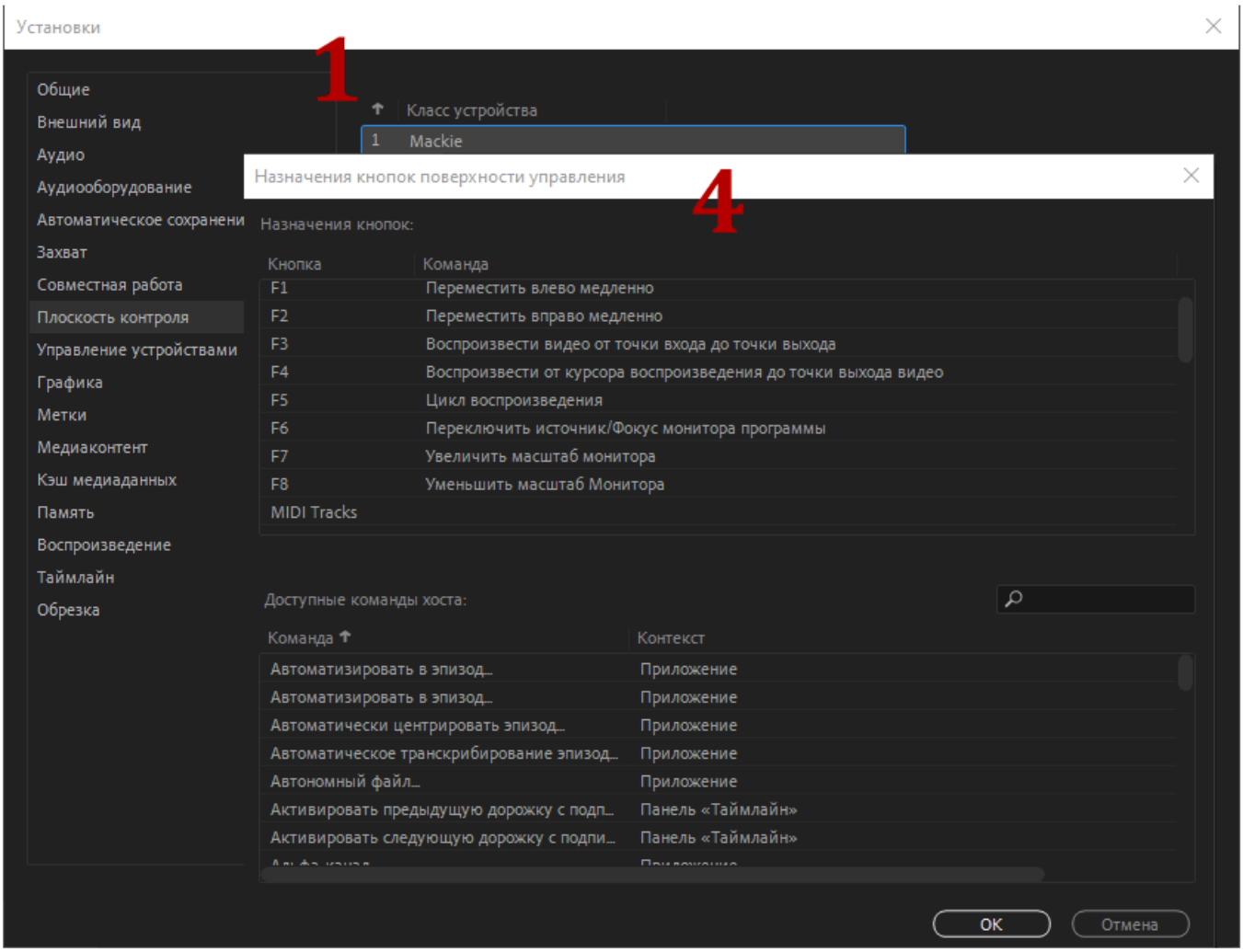
1.3.1. Premiere Pro Settings



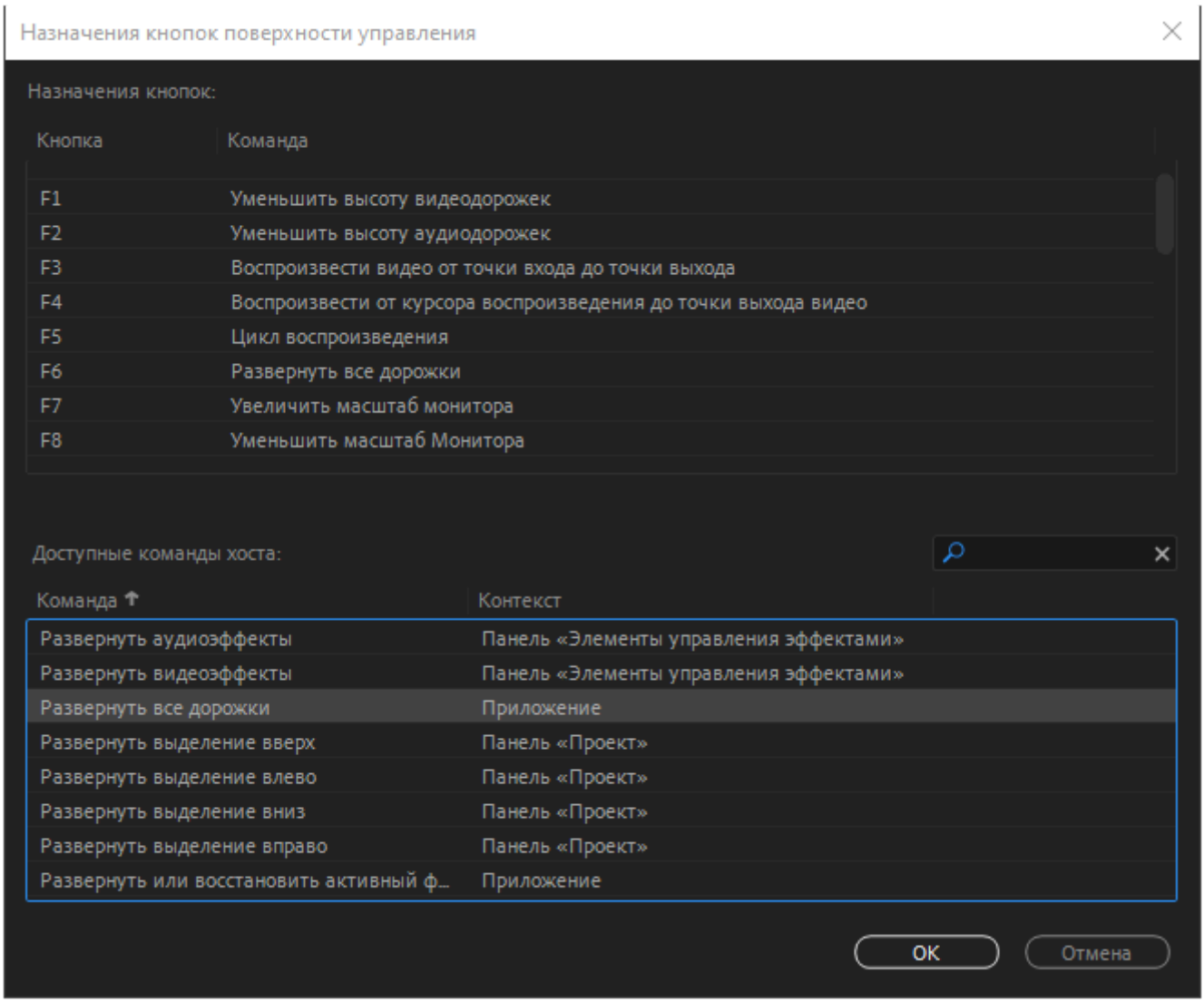
We select the purpose of the buttons in the **Premier PRO** application.



Now we need to assign actions to the functions bound to the controller buttons in the **Premier PRO** application.



The recommended choice of default controller settings, although you will most likely have your own preferences.



1.4. The configuration file settings

The configuration file has a popular Json format and you can use any text editor to change it. A working example config file is also included in the distribution. You can find it in the application installation folder, a copy of the configured configuration is located in the **docs** subfolder with the name **MidiController.cnf**.

The main parameters of the configuration file can be configured from the launched dialog by selecting the menu item **Start**. You can also adjust these settings directly while editing the file.

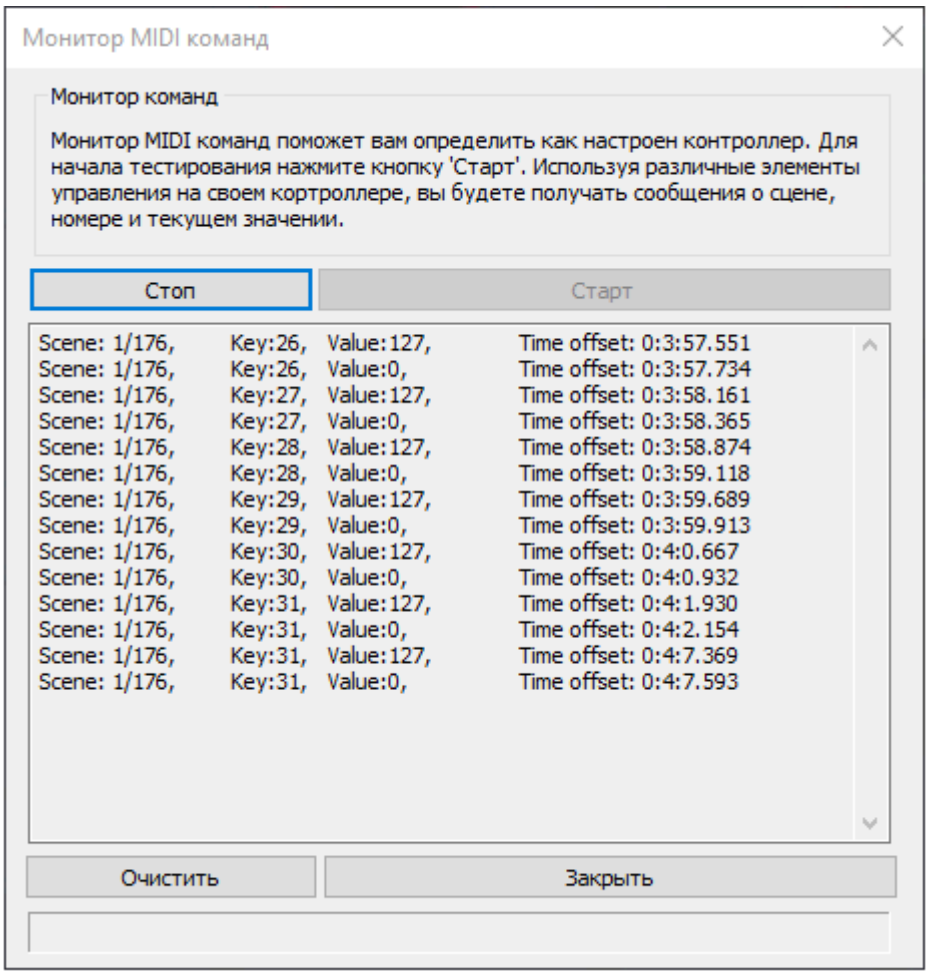
1.5. General configuration section

```
{
    // MIDI input port, EasyControl output
    "name":"W-FADER",
    // the name of the following configuration, in a normal scenario should be the
same as the file name
    "config":"MidiController",
    // the dialog will not wait for the Start button to be pressed, but will start
automatically
    "autostart":"true",
    // will not create virtual MIDI ports. You need to create the port yourself
    "manualport":"false",
    // creating a MIDI output proxy port for connecting other applications to the
controller
    "proxy":"true",
    // the interval between presses adjusts the "contact bounce"
    "btninterval":50,
    // long press interval adjusts the "long press" timeout
    "btnlonginterval":500,
    // 'Smart Home' settings block, to access the MQTT server
    "mqtt": {
        "host": "192.168.1.1",
        "login": "ctrl9",
        "pass": "12345",
        "prefix": "",
        "sslpsk": "",
        "certcapath": "",
        "port": 1883,
        "loglevel": 16,
        "isssl": "false",
        "selfsigned": "false"
    },
    "units":[... description of controller controls...]
}
```

The values of global parameters and their relationship are described in more detail in the section **Startup**.

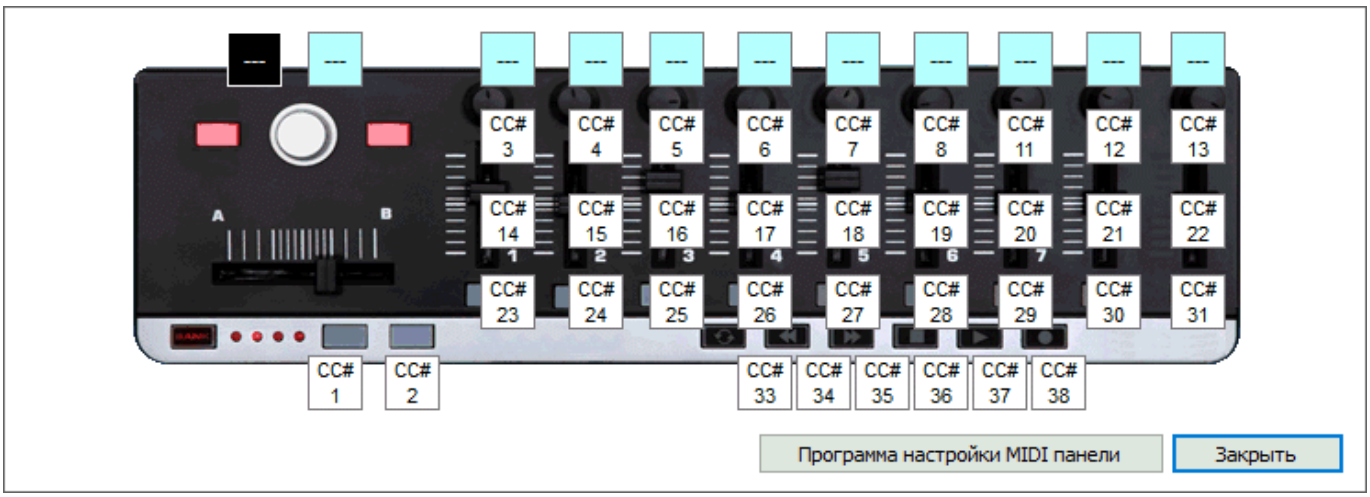
1.6. Control configuration section

You can check and compare the settings of the controller and the configuration file by selecting the **Monitor** item in the **MIDI-MT** menu.



The control description string looks like this:

```
{"scene":177,"id":22,"type":0,"target":0,"longtarget":255}
```



Scene (scene), the real Scene MIDI ID is specified in the configuration. The scene is the selected preset, there are 4 of them in the controller:

MIDI ID	Scene	note
176	1	no
177	2	no
178	3	no
179	4	no
192	1	for JOG control
193	2	for JOG control
194	3	for JOG control
211	4	for JOG control
243	4	for the top two buttons on the sides JOG
255	-	no scene selected

ID (id), MIDI control identifier, I hope everything is clear here.

MIDI ID	possible values
9	slider A-B, fixed value
10	encoder JOG, fixed value
64	right button from the encoder, fixed value
67	left button from the encoder, fixed value
0 - 170	any values in this range
170 - 254	used to mark scenes, etc

Type (type) of control:

No.	control type	enum
0	circular potentiometer	FADER
1	potentiometer "slider"	SLIDER
2	JOG, encoder	KNOB
3	button	BUTTON
4	button with toggle enabled	BUTTON TOGGLE
5	JOG, encoder, direction inversion	KNOB INVERSE
255	not defined	-

Destination (target,longtarget), destination of the control payload.

For buttons, the first value, **target** is a quick press, the second value, **longtarget** is a long press. If it is not necessary to use a long press, set the **longtarget** property to 255. For other types of controls, the second value is not processed, but needs to be set.

No.	payload
0	audio volume (master)
1	audio balance, panorama (master)
2	audio mute (master)
3	audio solo (master)
4	JOG - video frame by frame rewind
5 - 13	audio volume - channels 1-9
14 - 22	audio balance, panorama - channels 1-9
23 - 31	audio solo - channels 1-9
32 - 40	audio mute - channels 1-9
41 - 49	Audio Select - Channels 1-9
50 - 57	functions F1-F8, the standard mode of the button
58 - 65	functions F1-F8, works while the button is held down
66	Rewind
67	Forward
68	Stop
69	Play
70	Record
71	Up
72	Down
73	Left
74	Right
75	Zoom
76	Scrub
255	Element not selected

Also, the **target** value can indicate the relation to the group, as in the case of the `Audio sessions`, `Multimedia keys` and `Smart Home` groups. In this case, the value of **longtarget** determines the payload.

No.	group
252	module Smart Home
253	module Multimedia keys
254	module Audio sessions
255	элемент не выбран

1.7. Working example of control configuration


Settings Editor EasyControl MIDI Translator

```
{
    ....,
    /* настройка для сцены 2 */

    "units":[
        {"scene":177,"id":22,"type":0,"target":0,"longtarget":255},
        {"scene":177,"id":13,"type":1,"target":1,"longtarget":255},
        {"scene":177,"id":31,"type":3,"target":3,"longtarget":255},

        // A-B slider, disabled
        {"scene":177,"id":9,"type":255,"target":255,"longtarget":255},
        // jog encoder
        {"scene":177,"id":10,"type":2,"target":4,"longtarget":255},
        {"scene":177,"id":1,"type":3,"target":50,"longtarget":55},
        {"scene":177,"id":2,"type":3,"target":51,"longtarget":55},
        // right button next to jog encoder
        {"scene":177,"id":64,"type":3,"target":56,"longtarget":255},
        // left button next to jog encoder
        {"scene":177,"id":67,"type":3,"target":57,"longtarget":255},

        {"scene":177,"id":33,"type":3,"target":54,"longtarget":255},
        {"scene":177,"id":34,"type":3,"target":66,"longtarget":255},
        {"scene":177,"id":35,"type":3,"target":67,"longtarget":255},
        {"scene":177,"id":36,"type":3,"target":68,"longtarget":255},
        {"scene":177,"id":37,"type":3,"target":69,"longtarget":255},
        {"scene":177,"id":38,"type":3,"target":52,"longtarget":255},

        {"scene":177,"id":14,"type":1,"target":5,"longtarget":255},
        {"scene":177,"id":15,"type":1,"target":6,"longtarget":255},
        {"scene":177,"id":16,"type":1,"target":7,"longtarget":255},
        {"scene":177,"id":17,"type":1,"target":8,"longtarget":255},
        {"scene":177,"id":18,"type":1,"target":9,"longtarget":255},
        {"scene":177,"id":19,"type":1,"target":10,"longtarget":255},
        {"scene":177,"id":20,"type":1,"target":11,"longtarget":255},
        {"scene":177,"id":21,"type":1,"target":12,"longtarget":255},

        {"scene":177,"id":3,"type":1,"target":14,"longtarget":255},
        {"scene":177,"id":4,"type":1,"target":15,"longtarget":255},
        {"scene":177,"id":5,"type":1,"target":16,"longtarget":255},
        {"scene":177,"id":6,"type":1,"target":17,"longtarget":255},
        {"scene":177,"id":7,"type":1,"target":18,"longtarget":255},
        {"scene":177,"id":8,"type":1,"target":19,"longtarget":255},
        {"scene":177,"id":11,"type":1,"target":20,"longtarget":255},
        {"scene":177,"id":12,"type":1,"target":21,"longtarget":255},

        {"scene":177,"id":23,"type":3,"target":23,"longtarget":32},
        {"scene":177,"id":24,"type":3,"target":24,"longtarget":33},
        {"scene":177,"id":25,"type":3,"target":25,"longtarget":34},
```

```
        {"scene":177,"id":26,"type":3,"target":26,"longtarget":35},
        {"scene":177,"id":27,"type":3,"target":27,"longtarget":36},
        {"scene":177,"id":28,"type":3,"target":28,"longtarget":37},
        {"scene":177,"id":29,"type":3,"target":29,"longtarget":38},
        {"scene":177,"id":30,"type":3,"target":30,"longtarget":39},
    ]
}
```

1.8. Audio Mixer

Additionally, it is possible to manage **audio sessions** of running applications. Control can occur both from a MIDI keyboard and from the built-in sound control panel.

The audio sessions control panel has several themes. You can choose the design of the panel according to your taste and the theme of your desktop. Also, there is a customizable theme, in which you can change all the colors as you wish.

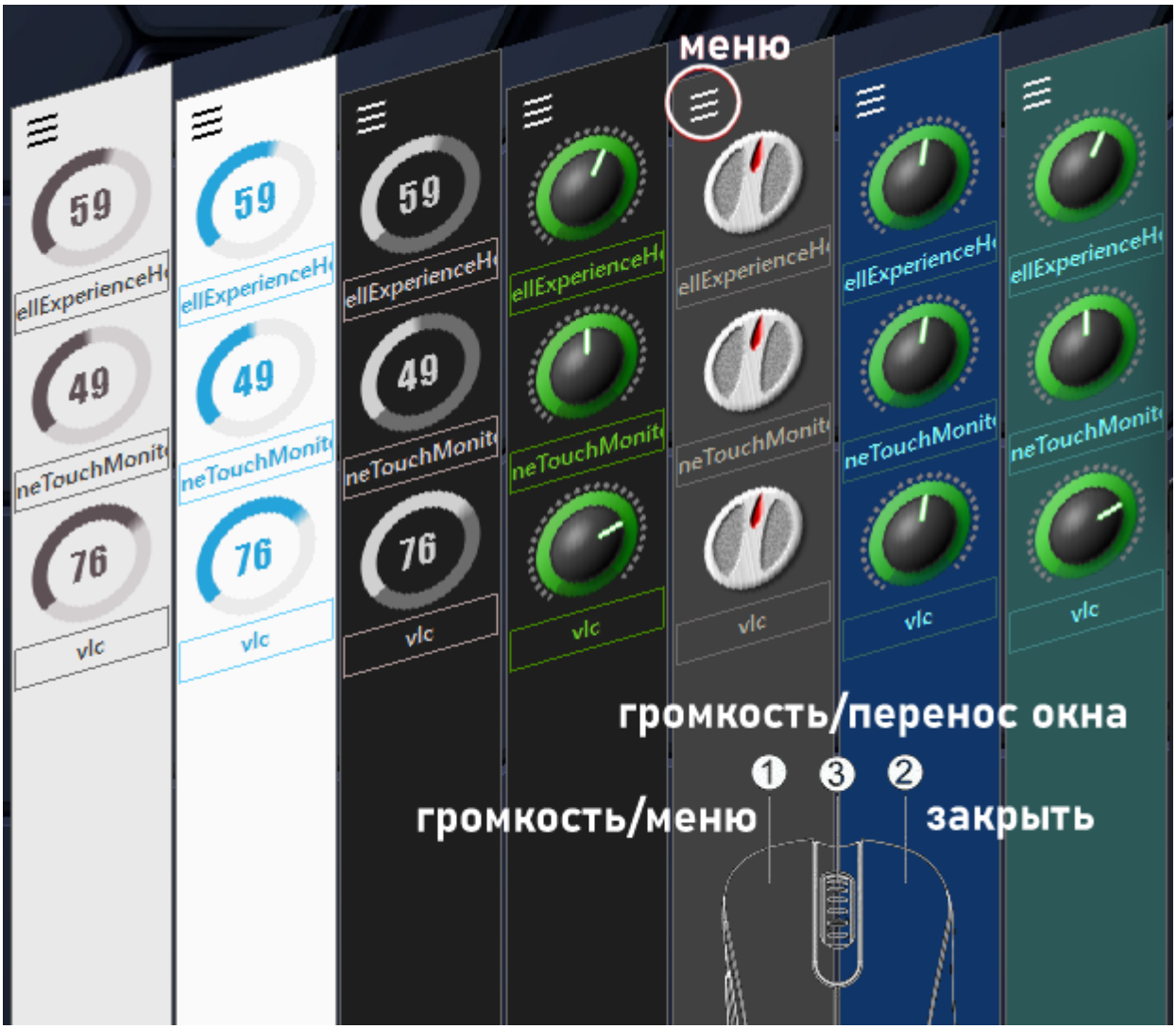
1.8.1. The following panel themes are available:

- `Light`,
- `Dark`,
- `Metro`,
- `Retro`,
- `Modern`,
- `Customizable`

Panel placement on the screen has four fixed initial positions:

- `vertical-right`,
- `vertical-left`,
- `horizontal-top`,
- `horizontal-bottom`

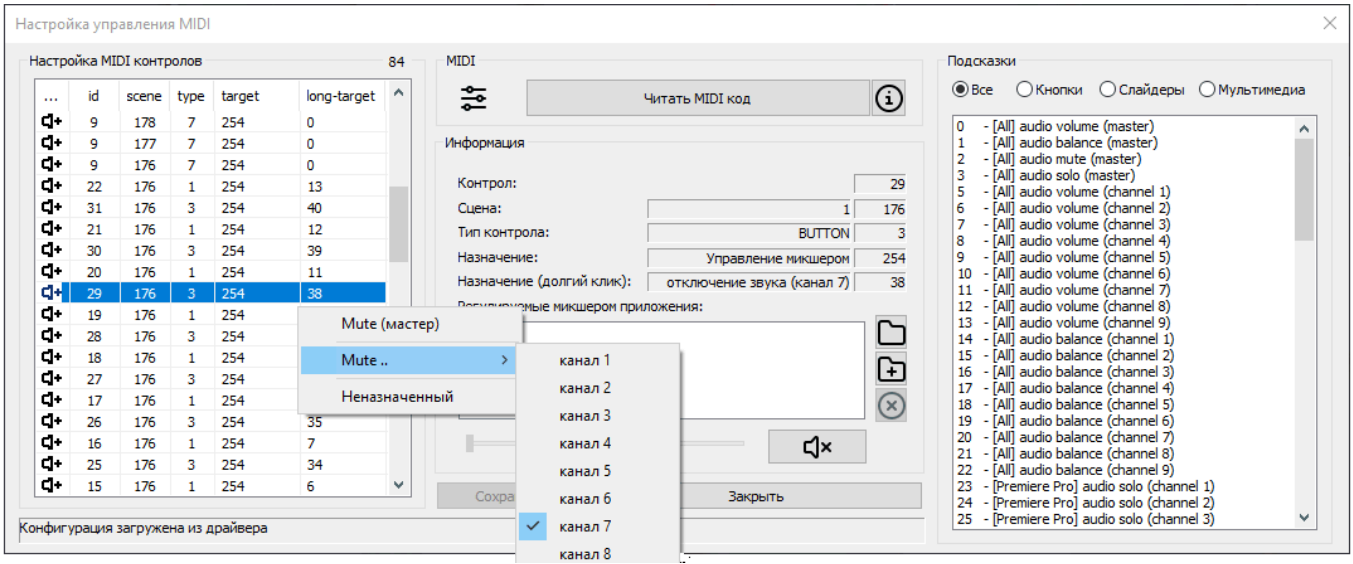
You can move the panel in the plane of its location at your own discretion. In other words, panels in a horizontal position can be moved vertically, and panels in a vertical position can be moved horizontally. On the panel, you can adjust the volume and mute for each session. Adjustment is possible both from the MIDI keyboard and the mouse, using the built-in audio panel.



To control audio sessions from a MIDI keyboard, there is no need to call up the panel. The panel is only needed to adjust the sound with the mouse, or to visualize the status of ongoing audio sessions.

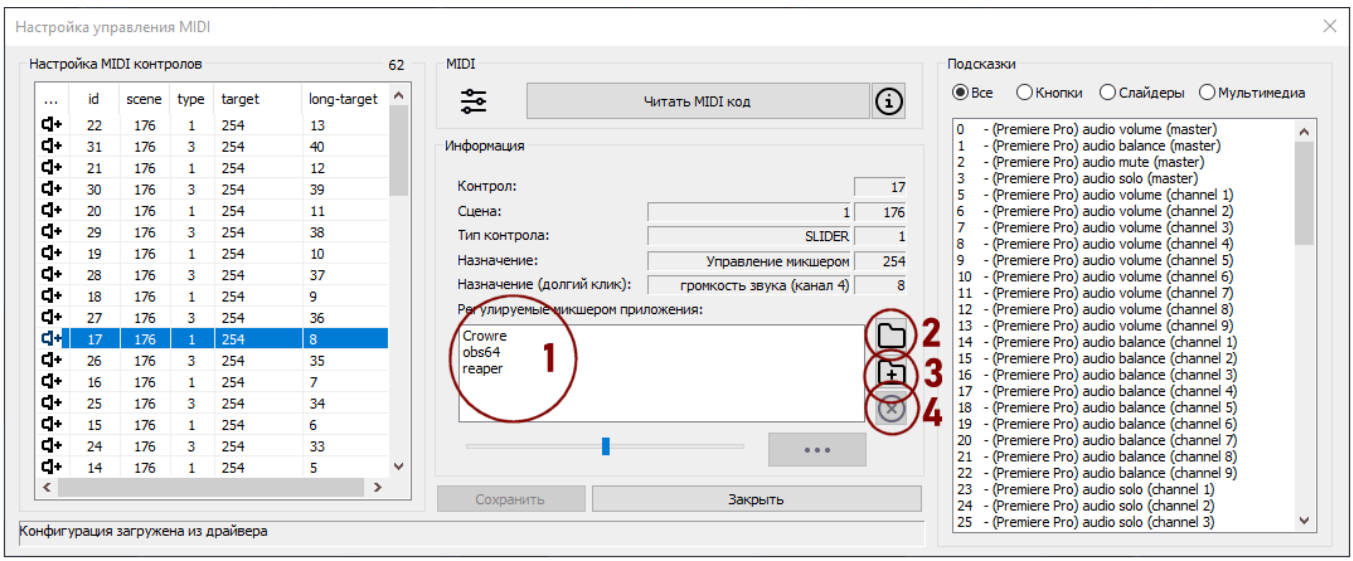
1.8.2. Setting

The setting is done in the `controls editor`. To add an application whose volume you want to control, you need to add a new `control` in the editor or select an existing one. Edit it. In the `target` field, you must specify the identifier of the mixer, this is the number `254`, or the multimedia identifier of the button controller, this is the number `253`. In the `long-target` field, you must specify the purpose of the `control`, what it will regulate. On the left side of the editor, there are hints to help you select the correct value..



After that, you must add the full path to the managed program to the list of applications. The editor itself will turn it into the required format during processing, this value will be the application identifier. You can add an application through the file selection dialog by specifying the desired executable file on the disk, or through the `downloaded applications` button. In the latter case, all running applications that have an open audio session will be added. Extras can be removed.

The number of managed applications per `control` is unlimited. But we must remember that if several applications specified by one `control` are loaded, then only the last one will adjust the sound.



Especially relevant, using the built-in mixer to control sound in **Windows 11**, where the usual sound controls have been removed, and access to existing ones is associated with a long journey through the settings menu.

2. Smart home device control

```
MQTT - Message Queuing Telemetry Transport
```

You can integrate a **MIDI keyboard** into your smart home system to control various devices. Any **Smart Home** control environments that are based on the MQTT exchange protocol and include an MQTT server are supported.

In the Smart Home system, the MQTT protocol must be at least level `5.0`, it is possible to work with earlier versions, but stability and full performance for all commands are not guaranteed.

We recommend the following compatible smart home control solutions:

- any system based on MQTT broker `mosquitto`,
- `Home Assistant`,
- `openHAB`,
- `MajorDoMo`,
- `ioBroker`,
- `Domoticz`,
- `MyController`,
- `IntraHouse`,
- and so on..

2.1. MQTT topics (topics)

MIDI-MT uses the following MQTT topic structure:

```
/sensor/<login for MQTT server>/<control identifier>/<value name>
```

Example of used topics. Let's assume that your login for the MQTT server is `ectrl9`, and the identifier of the control that smoothly adjusts the lighting is `av1` and the control to turn on or off the light has the identifier `b11`. In this case, control commands will be sent to the following topics:

```
/sensor/ectrl9/av1/level = 0 to 127
/sensor/ectrl9/b11/onoff = 0 to 1
```

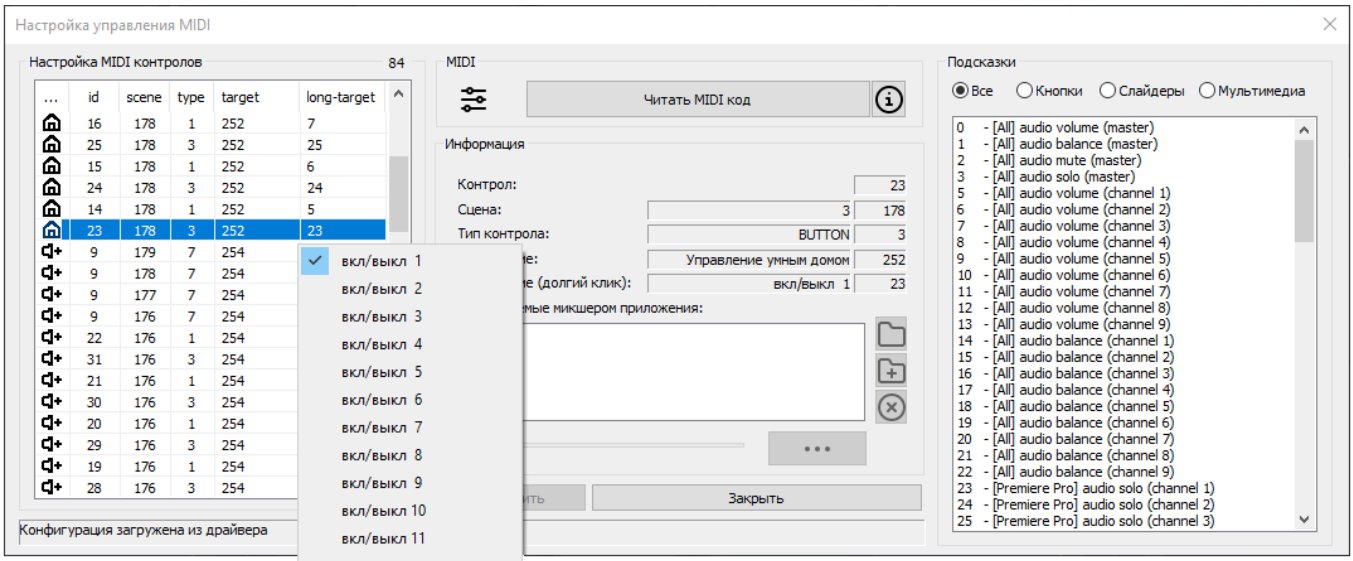
The name of the control is set automatically, in accordance with the control ID selected in the editor. It can be obtained by referring to the topic:

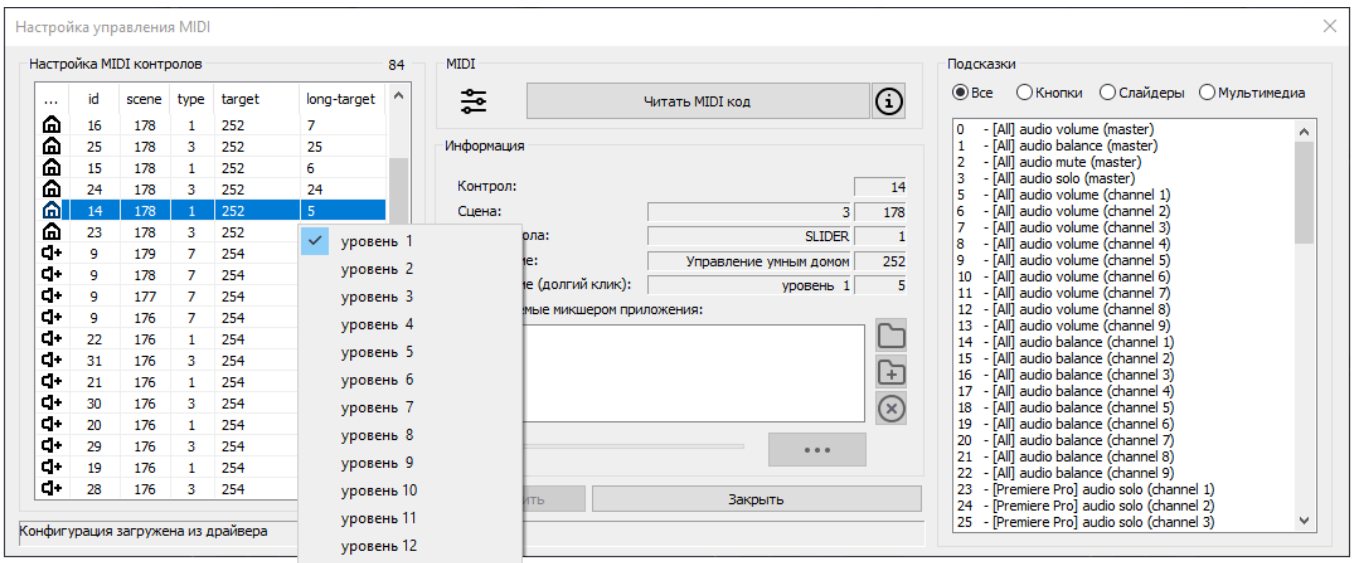
```
/sensor/ectrl9/av1/title = "level 1"
/sensor/ectrl9/b11/title = "on/off 1"
```

For some scenarios, information about the status of a **MIDI keyboard** connection will be useful. You can check its current state by referring to the topic:

```
/sensor/ectrl9/state = 0 to 1
```

The necessary controls can be configured in the **configuration editor**. MQTT connection settings are stored in the configuration, so it is possible to have an unlimited number of configured configurations that can be loaded as needed.





If you are poorly versed in the MQTT protocol, or in the Smart Home device, in this case, I recommend reading the book:

- MQTT Essentials E-Book,
  or there is an online version:
- MQTT Essentials and MQTT версия 5 Essentials.

2.2. Setting up an MQTT connection

2.2.1. SSL/TLS - certificate setup

The certificate file, the path to which you must specify in the configuration, must consist at least of the CA (Certificate Authority) certificate that signed the certificate of your MQTT server. If you have a chain of required certificates, then you need to place the root certificate in a separate folder, and add the rest of the necessary certificates to it. For example, the root certificate of `Let's Encrypt` is named `R3`. If you are using a certificate from this company, then you just need to specify the path to the `R3.cert` or `R3.pem` file in PEM format.

In the case of a self-signed certificate, it is enough to indicate it, while checking the box above the item of the same name. In this case, the MQTT server certificate will be the same as the CA certificate.

To enable SSL/TLS, check the box above the corresponding item. Use these options only if you want to encrypt communication between client and server. On a local network, this is usually not worth doing. Since this increases the load on both the client and the server.

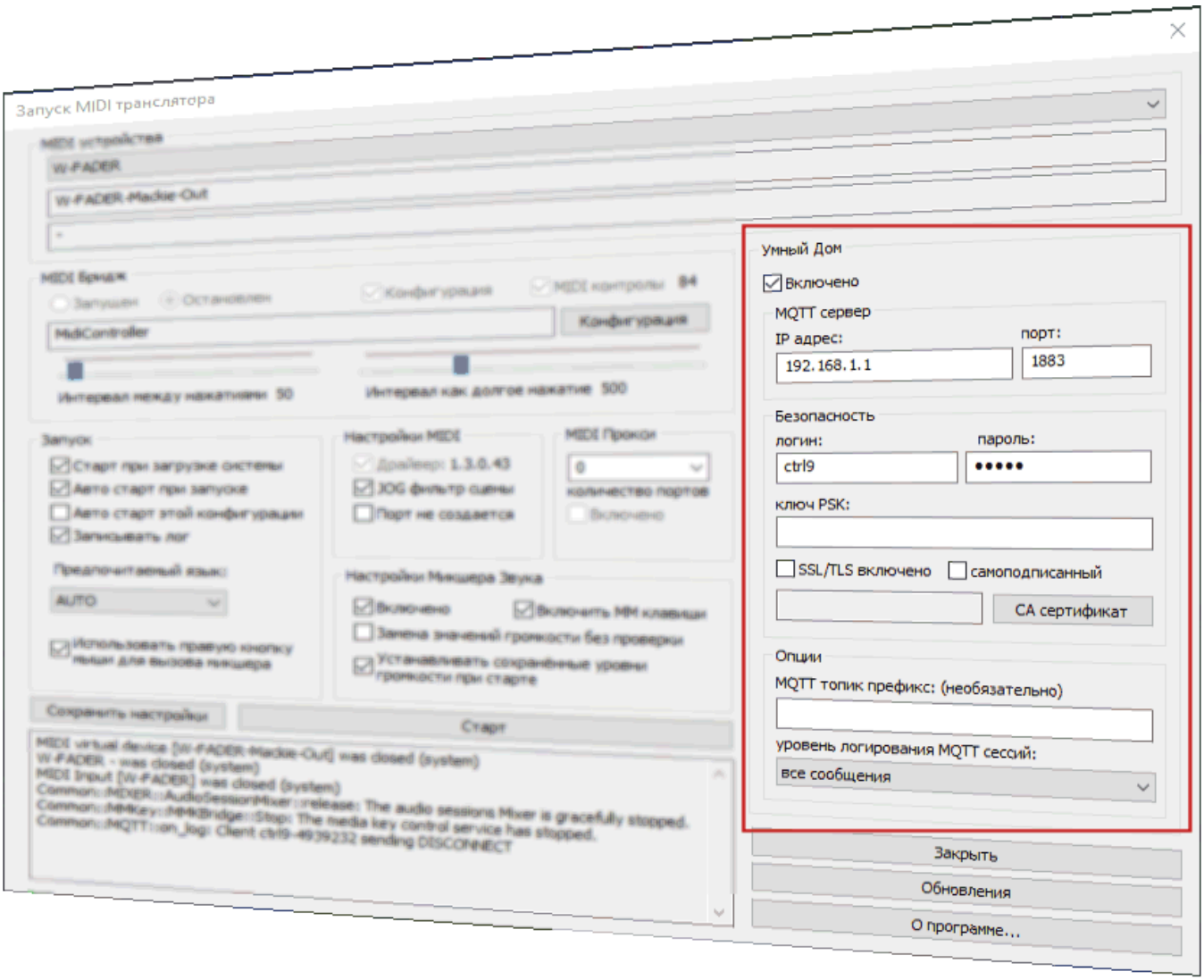
2.2.2. PSK - setting keys

Working with PSK keys is not compatible with SSL/TLS settings. The connection can work only in one of the specified modes, when all values are filled in, priority is given to the connection based on PSK keys. They use PSK keys, as a rule, when working with external servers, the authorization of which is configured through their own Web interface.
For example `AdaFruit MQTT` or `HiveMQ Public MQTT Broker`.

The whole setup comes down to getting the key, and copying it into the appropriate field. The second parameter of the PSK key binding is your login to the MQTT server, since you have already filled it out above, these are all the settings that can be applied.

2.2.3. Other settings

The remaining settings, such as: server IP address, its port, login, password and log level, are obvious and do not require explanation.



2.2.4. Configuration file

The settings in the config file look like this:

```
{
    ...
    // 'Smart Home' settings block, to access the MQTT server
    "mqtt": {
        "host": "192.168.1.1",
        "login": "ctrl9",
        "pass": "12345",
        "prefix": "",
        "sslpsk": "",
        "certcapath": "",
        "port": 1883,
        "loglevel": 16,
        "isssl": "false",
        "selfsigned": "false"
    },
    ...
}
```

From the recommendations, the logging level can help in debugging associated with the first connection to the server. In the future, it is better to turn it off completely, as this creates an additional load on the server, since it broadcasts all the information about its own actions. The log level has nothing to do with local messages.

2.2.5. Example log file

```
[07-25 -> 16:58:38] The selected configuration does not support auto start.
[07-25 -> 16:58:38] Common::MIDI::MidiBridge::Start: new configuration version
detected..
[07-25 -> 16:58:38] Common::MIDI::MidiBridge::Start: configuration already loaded..
[07-25 -> 16:58:38] MIDI Input device found: W-FADER/0
[07-25 -> 16:58:38] W-FADER - was opened (system)
[07-25 -> 16:58:38] MIDI Input device open: W-FADER/0
[07-25 -> 16:58:38] MIDI virtual device driver version [1.3.0.43]
[07-25 -> 16:58:38] MIDI virtual device [W-FADER-Mackie-Out]: OK
[07-25 -> 16:58:38] MIDI Output device open: W-FADER-Mackie-Out/1
[07-25 -> 16:58:38] Common::MIXER::AudioSessionMixer::init: Mixer audio sessions
started, at the time of start, there are 3 running programs available for regulation.
[07-25 -> 16:58:38] Common::MMKey::MMKBridge::Get: The media key control service is
running.
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending CONNECT
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending PUBLISH (d0,
q0, r1, m1, 'sensor/ctrl9/state', ... (1 bytes))
[07-25 -> 16:58:38] Common::MQTT::SmartHome::Start: The Smart-Home control service is
running.
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending PUBLISH (d0,
q0, r1, m2, 'sensor/ctrl9/b37/title', ... (9 bytes))
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending PUBLISH (d0,
q0, r1, m3, 'sensor/ctrl9/av1/title', ... (7 bytes))
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending PUBLISH (d0,
q0, r1, m4, 'sensor/ctrl9/av2/title', ... (7 bytes))
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending PUBLISH (d0,
q0, r1, m5, 'sensor/ctrl9/av3/title', ... (7 bytes))
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending PUBLISH (d0,
q0, r1, m6, 'sensor/ctrl9/av4/title', ... (7 bytes))
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending PUBLISH (d0,
q0, r1, m7, 'sensor/ctrl9/av5/title', ... (7 bytes))
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending PUBLISH (d0,
q0, r1, m8, 'sensor/ctrl9/av6/title', ... (7 bytes))
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending PUBLISH (d0,
q0, r1, m9, 'sensor/ctrl9/av7/title', ... (7 bytes))
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending PUBLISH (d0,
q0, r1, m10, 'sensor/ctrl9/av8/title', ... (7 bytes))
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending PUBLISH (d0,
q0, r1, m11, 'sensor/ctrl9/b11/title', ... (8 bytes))
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending PUBLISH (d0,
q0, r1, m12, 'sensor/ctrl9/b12/title', ... (8 bytes))
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending PUBLISH (d0,
q0, r1, m13, 'sensor/ctrl9/b13/title', ... (8 bytes))
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending PUBLISH (d0,
q0, r1, m14, 'sensor/ctrl9/b14/title', ... (8 bytes))
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending PUBLISH (d0,
q0, r1, m15, 'sensor/ctrl9/b15/title', ... (8 bytes))
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending PUBLISH (d0,
```

```
q0, r1, m16, 'sensor/ctrl9/b16/title', ... (8 bytes))
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending PUBLISH (d0,
q0, r1, m17, 'sensor/ctrl9/b17/title', ... (8 bytes))
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending PUBLISH (d0,
q0, r1, m18, 'sensor/ctrl9/b18/title', ... (8 bytes))
[07-25 -> 16:58:38] Common::MQTT::on_log: Client ctrl9-4687392 sending PUBLISH (d0,
q0, r1, m19, 'sensor/ctrl9/b32/title', ... (9 bytes))
```