



User documentation

FLEXGUI

Content

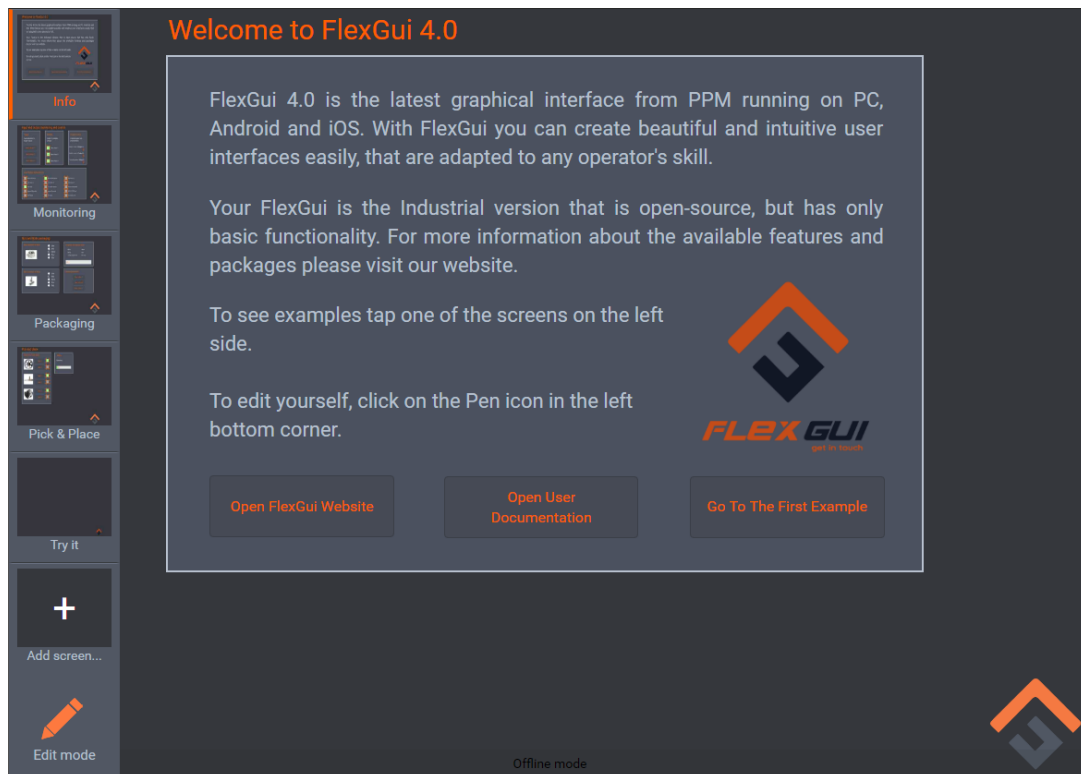
1	BeltTechnology introduction	4
1.1	Screen Belt.....	Error! Bookmark not defined.
1.1.1	Editing screen properties	6
1.2	Edit Belt	Error! Bookmark not defined.
1.2.1	Elements of Edit Belt	7
1.3	Delete Belt	Error! Bookmark not defined.
1.4	Properties Belt.....	Error! Bookmark not defined.
1.4.1	Image Explorer Window	8
1.4.2	Saving Changes	8
1.4.3	Arranging Fidgets.....	9
1.5	Fidget Belt.....	Error! Bookmark not defined.
1.5.1	Fidgets	9
2	Modal windows	13
2.1	Help	14
2.2	Settings	14
2.2.1	General	14
2.2.2	Init script.....	15
2.2.3	Nodes.....	15
2.2.4	Connection settings.....	Error! Bookmark not defined.
2.2.5	Messenger	16
2.2.6	Language.....	17
2.2.7	Project operations	17
2.2.8	Diagnostics.....	18
2.2.9	Backup	21
2.3	Properties	21
2.4	Popup messages	22
2.5	Add screen.....	23
2.6	Login	23
2.7	Connections.....	23
2.8	Timers.....	23
2.9	User Mode	25
	Please note: The user name is “admin”	25
2.10	Enterprise	25
3	Moving Fidgets	26
4	Factory designer	27

5	Programming FlexGui	29
5.1	FlexGui Script Editor	29
5.2	JavaScript function reference.....	29
5.2.1	#autoInit	29
5.2.2	#setScreenByName.....	30
5.2.3	#setScreenByIndex	30
5.2.4	#message	30
5.2.5	@friendlyName	30
5.3	JavaScript basics	30
5.3.1	Statements	30
5.3.2	Comments	31
5.3.3	Variables	31
5.3.4	Operators.....	31
5.3.5	Conditional statements	32
5.3.6	Loops	33
6	ROS-I connection	34
6.1	Overview.....	34
6.2	Publishing a Topic.....	34
7	Direct connection	36
7.1	Variables and functions	37
7.1.1	Variables	37
7.1.2	Subscribe	38
7.1.3	Unsubscribe	38
7.1.4	Change value	38
8	The ExpertSite	39
8.1	Expert Site Fidgets	40
8.2	Factories	41
8.3	Manage screen	42
8.3.1	User management	43
8.4	Messenger	44
8.4.1	Contacts.....	44
8.5	Alarm subscriptions.....	45
8.6	Accounts	46
9	FlexGui Node	47
	Configuration and setup.....	47
9.1	High Speed Camera Proxy	47

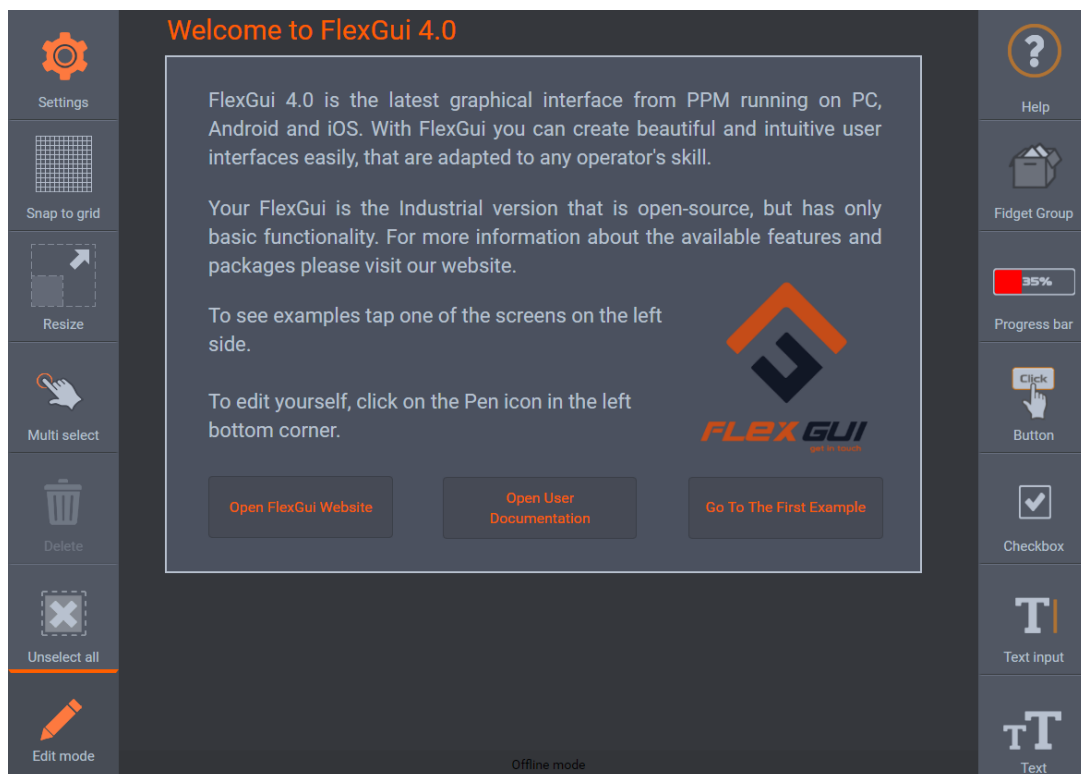
Configuration and setup.....	47
Interface	48
9.2 Alarms.....	48
Configuration and setup.....	48

1 BeltTechnology introduction

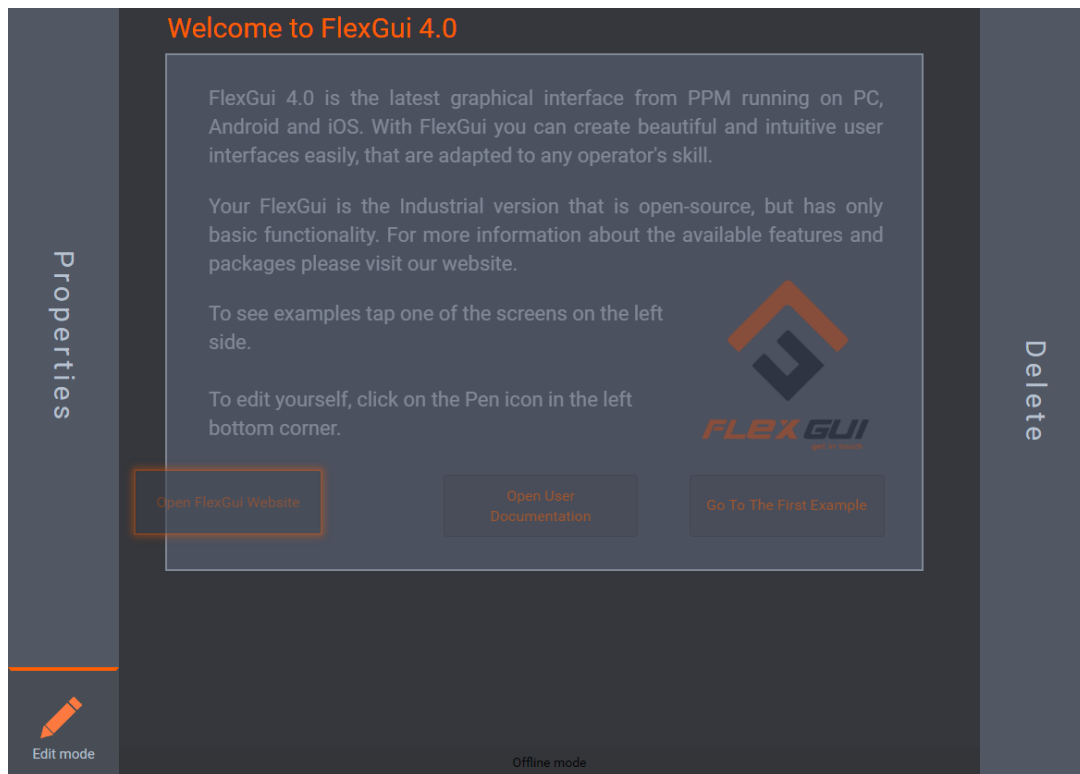
FlexGui contains 5 different belts for a touch, drag&drop optimized user experience.



1 The Screen Belt to access the screens in the project. A screen can contain many Fidgets, it serves a well-defined a purpose.



2 The Edit and Fidget Belts, that appear when the edit mode is turned on using the pencil icon in the left bottom corner. The Edit Belt contains useful editing features, while the Fidget Belt contains the usable Fidgets that the user can drag and drop.



3 The Properties and Delete Belts, that appear when the user drags a Fidget in edit mode. They serve as goals of dropping to edit or remove a Fidget.

Let's see these items in more details.

1.1 Screen Belt

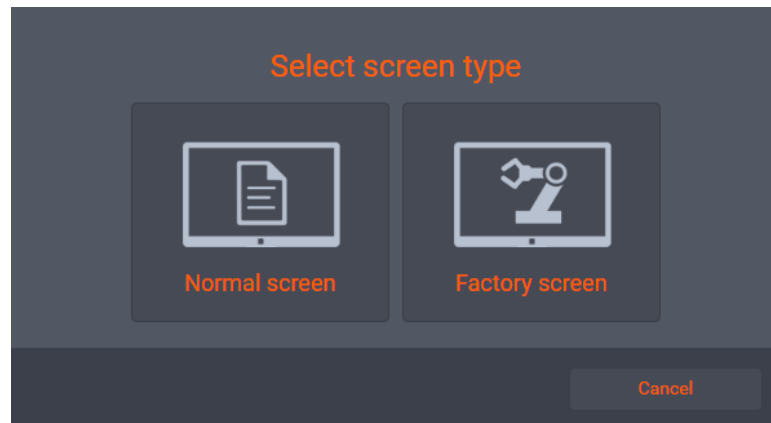
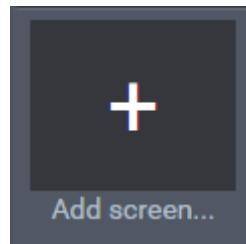
Screen Belt is on the left side of the screen if the user is not in Edit Mode.

If the user is logged in, on the bottom of the Screen Belt there is the Edit Mode switch.

If the user is in Guest mode (not logged in), on the bottom of this belt there is the Login button, which opens the login screen:

On the login screen the user can choose to log in and gain elevated rights, or remain a guest who can use only a subset of features.

The Screen Belt contains all of the available screens in the project. The logged-in users can add new screens by clicking the Add screen button and selecting the type of the screen on the popup window:



Factory screens are for designing the 2D layout of the user's factory to have an overview, Normal screens are used to observe or control one or more devices in the factory.

There are specific types of Fidgets for **Factory** and **Normal** screens on their Fidget Belts.

1.1.1 Editing screen properties

Long pressing the selected screen will open the Screen properties window.

A dialog box titled "Properties of Info" with a close button (X) in the top right corner. It contains several fields: "Has screen belt" (a dropdown menu), "Logo" (a text field with the value "iconService.get('images/logo_small.png')"), "Logo position" (a dropdown menu with "Bottom - Right"), "Logo width" (a text field with "100" and a hint "[Long tap to open variables]"), "Name" (a text field with "Info" and a hint "[Long tap to open variables]"), "Background type" (a dropdown menu with "Color"), and "Background color" (a text field). There are "Select" and "Pick..." buttons next to the "Logo" and "Background color" fields respectively. At the bottom, there are buttons for "Duplicate", "Delete", "Move up", "Move down", "Save", and "Cancel".

Screens have name and background properties (for normal screens an image, for factory screens a color) and the position on the Screen Belt. The Screen Belt can be switched off on all screens, so the screen has more space, or the integrator can decide to limit the user's available screens to a selected set.

Both properties can be set up here and it is also possible to delete and move the selected screen.

1.2 Edit Belt

The Edit Belt is on the left side of the screen in **Edit Mode**.

1.2.1 Elements of the Edit Belt

- **Settings:** opens the settings window
- **Snap to grid:** shows a grid and snaps Fidgets to this grid while dragging
- **Resize:** enables drag resize
- **Rotate (only on Factory screens):** enables drag rotate on factory screens
- **Multi select:** enables multiple Fidget selection. ***Please note**, that the Properties Belt is disabled while moving more than one Fidget.*
- **Delete:** Deletes all selected Fidgets
- **Unselect all:** removes the current selection
- **Undo:** undo last action
- **Redo:** redo last action
- **Copy:** copies the selected Fidgets to the clipboard
- **Cut:** copies the selected Fidgets to the clipboard and removes them from the screen
- **Paste:** pastes Fidgets from the clipboard (only available if one or more items are on the clipboard)

1.3 Delete Belt

The Delete Belt is on the right side of the screen while dragging one or more Fidgets in **Edit Mode**. The user can drop the Fidgets on this belt to remove them from the screen.

1.4 Properties Belt

Dragging Fidgets to Properties Belt opens the property window.

Properties of Button

Id: fidget_a6ea0458-0152-21a5-d343-c20b0b3daeca

Color: [Pick...]

Enabled: true [Long tap to open variables]

Font: inherit

Font size: 16 [Long tap to open variables]

Top: 453 [Long tap to open variables]

Left: 45 [Long tap to open variables]

Name: [Long tap to open variables]

Text: Open FlexGui Website [Long tap to open variables]

Width: 200 [Long tap to open variables]

Height: 67 [Long tap to open variables]

On click:

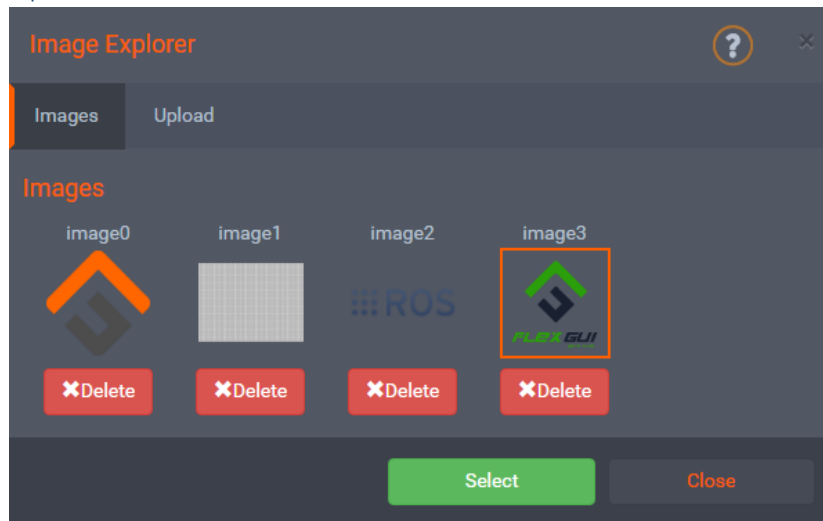
```
1 window.open("https://www.pgm.no/flexgui4-home", "_system");
```

[Send to back] [Send backward] [Bring to front] [Bring forward] [Save] [Cancel]

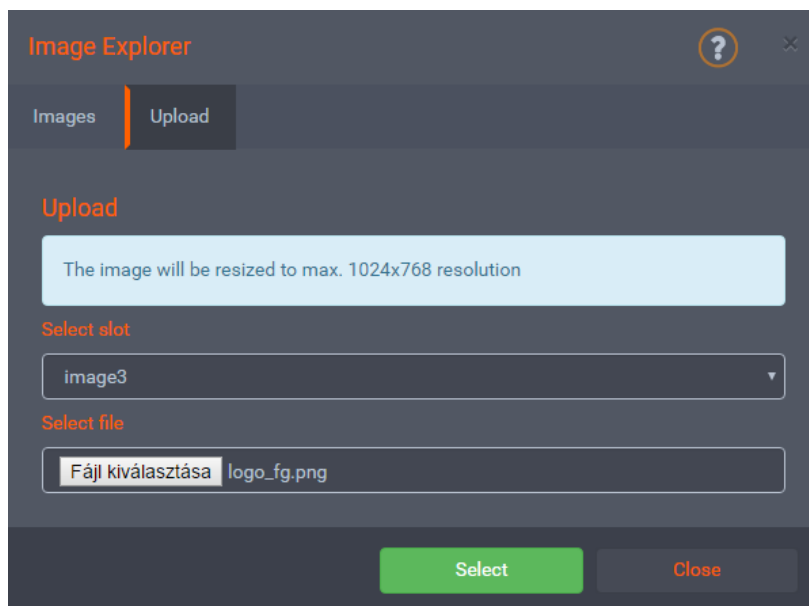
The elements of the property windows depend on the type of the Fidget. You can read more about the available properties in the next chapter. The type of a property editor field depends on the type of the

property (e.g. for images we can open an Image Explorer window and add/delete images on the backend and use as a source for a Fidget).

1.4.1 Image Explorer Window



On the picture above you can see the currently available pictures stored on the backend in ROSparams or local/network storage. It is possible to delete them or overwrite them by uploading.



1.4.1.1 Upload steps:

1. **Select slot:** on the backend there are 20 available slots for images, the user has to select which slot should be used.
2. **Select file:** the user has to select an image, which will be resized and uploaded to the server. After the upload, the image will be automatically selected on the Images tab.

1.4.2 Saving Changes

Save: save the current setup and close the property window

Cancel: restore the previous setup and close the property window

1.4.3 Arranging Fidgets

Fidgets can be arranged by these four buttons:

Send to back: go to the lowest layer of the screen

Send backward: go one layer down

Bring to front: go to the topmost layer of the screen

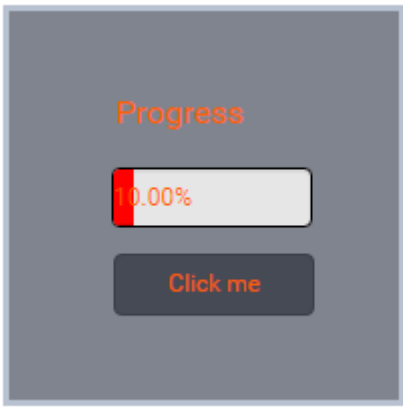

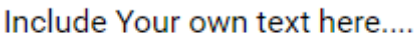
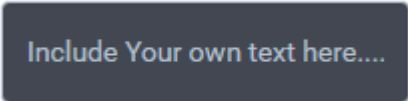
Bring forward: go one layer up

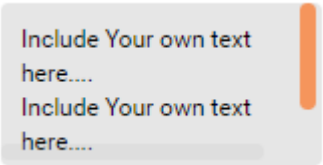

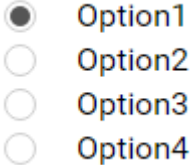
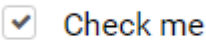



1.5 Fidget Belt


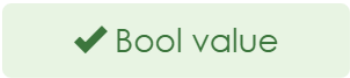




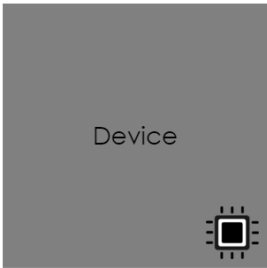
The Fidget Belt is on the right side of the screen in Edit Mode. The user can add new Fidgets to the current screen with drag & drop. To grab a Fidget from the belt, long tap / click has to be used. Releasing the tap / click adds the Fidget to the screen.

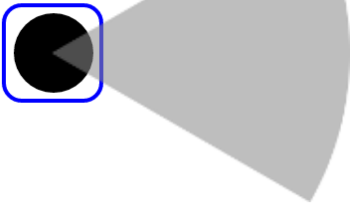


1.5.1 Fidgets

Every Fidget has 5 default properties: width, height, top, left and enabled. With these properties, the user can set the size and the position of the Fidget. The position of a Fidget is calculated from the top-left corner of its parent: a Fidget group or the screen. With the enabled property, it is possible to prevent any On Click action on a Fidget, thus no mouse actions are allowed.

Name	Fidget	Properties	Normal	Factory
Fidget group		Color Opacity On click Border color Border Width	x	x
Button		Text On click Font Font size Font color	x	
Text		Font Font size Color Text On click Text align	x	x
Text input		Text On click	x	

Scrollable text		Text On click Text align Font Font size Color	x	x
Progress		Value Color On click Font color	x	
Radio button		Options Font Font size Font color Value On click	x	
Checkbox		Text Value Font Font size Font color On click	x	
Image		Scale Image On click	x	
Gauge		Value Color Minimum Maximum Angle arc Angle offset	x	
Indicator lamp		Blinking Blink period On/off color Text Font color On click	x	

Messenger		On click	x	x
Bool		Value Text Font Font size Font color On click	x	
Slider		Minimum Maximum Value Step Precision On click	x	
Camera image		Source Scale On Click	x	
Road		Color Angle On click		x
Fence		Color Angle On click		x
Device		Font color Color Text Screen link Type (Robot/PLC)		x

Camera		Angle Text Border color Font color Color Screen link On click		x
End of way		Screen link Color Font color Angle Text On click		x
Generic obstacle		Color Font color Angle Text On click		x

2 Handling

FlexGui is designed to be used with touch and non-touch screen devices as well. You can use your fingers on a touch screen device to control FlexGui, or you can use your mouse and keyboard combination for easier editing and assembling of projects.

It is also possible to use keyboard shortcuts for easier handling with your PC.

Please note: keyboard shortcuts are part of the Shortcut addon. For more information please visit our [website](#).

2.1 Keyboard shortcuts

2.1.1 Edit mode:

- **Ctrl + C** = copy
- **Ctrl + X** = cut
- **Ctrl + V** = paste
- **Ctrl + A** = select all
- **Ctrl + S** = download current project
- **Ctrl + Z** = undo
- **Ctrl + Shift + Z** = redo
- **Delete** = delete
- **Pressing Shift only** = the resize tool will be active on selected Fidget
- **Releasing Shift only** = the resize tool will be inactive
- **Ctrl + Left Mouse Button** = Multiple select
- **Arrow keys** = move Fidget with 1px
- **Ctrl + Arrow keys** = move Fidget with the set up gridsize (Settings / General)

2.1.2 Playback mode (when Screen Belt is available)

- **Ctrl + I** = new screen


2.1.3 Both edit mode and playback mode

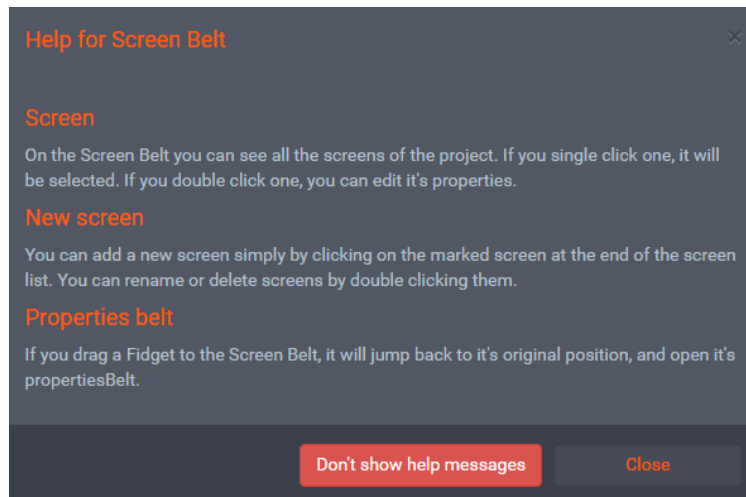
- **Alt + Down** = next screen (or first if the last one is active)
- **Alt + Up** = previous screen (or last if the first one is active)
- **Ctrl + E** = switch between edit mode/playback mode
- **Esc** =
 - Unselect all if some Fidgets are selected in edit mode
 - Cancel movement when dragging a Fidget in edit mode. The Fidget moves back to the original position regardless of holding or releasing LMB
- **Enter** =
 - Open properties when Fidget(s) are selected in edit mode
 - Open properties when a screen is selected in playback mode

3 Modal windows

3.1 Help

On-screen contextual help is available and can be enabled / disabled in the settings window.

The user can get instant help by pressing one of the  -icons anywhere in FlexGui to get contextual help.



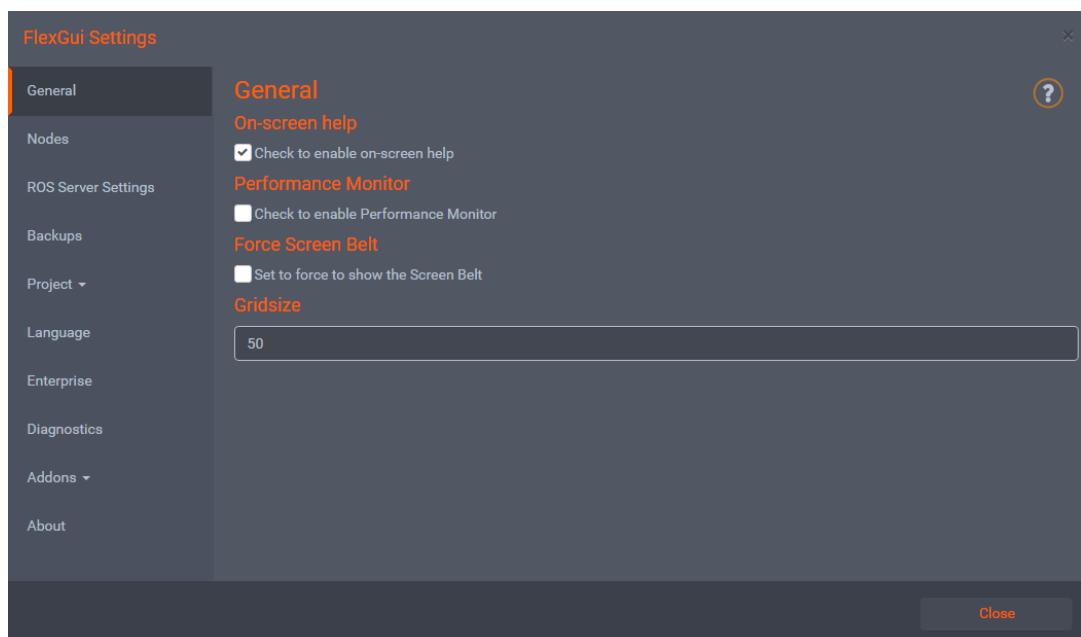
3.2 Settings

To open the settings window, the user has to enable **Edit Mode** and press the settings on the top of the Edit Belt.

3.2.1 General

General settings of FlexGui:

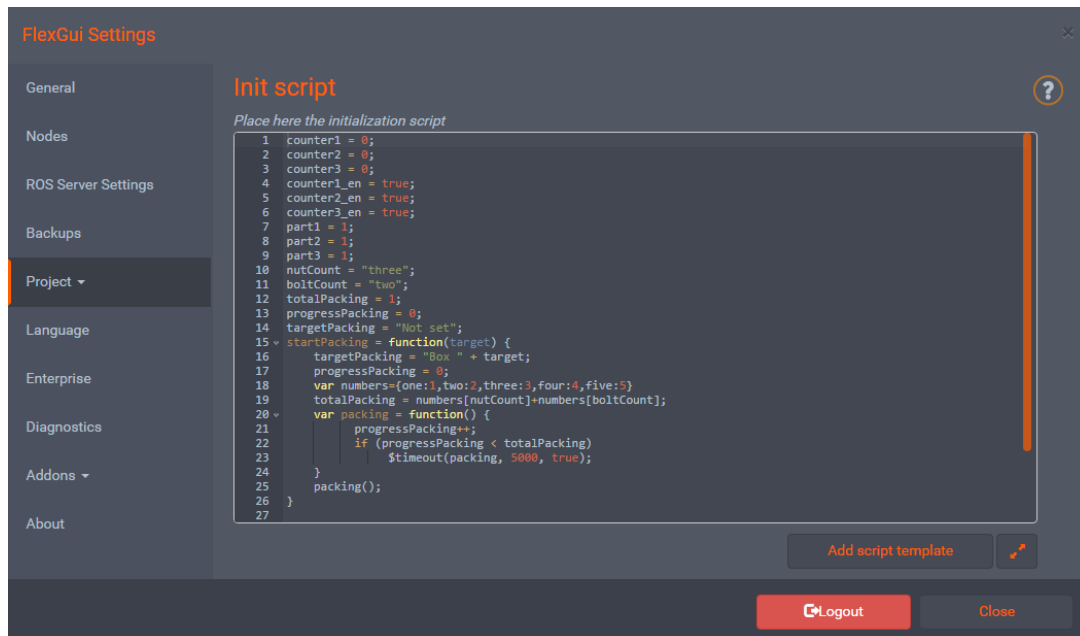
- Screen scaling (zoom on mobile devices),
- On-screen help on and off,
- Performance Monitor on and off,
- Grid size for edit mode.



3.2.2 Init script

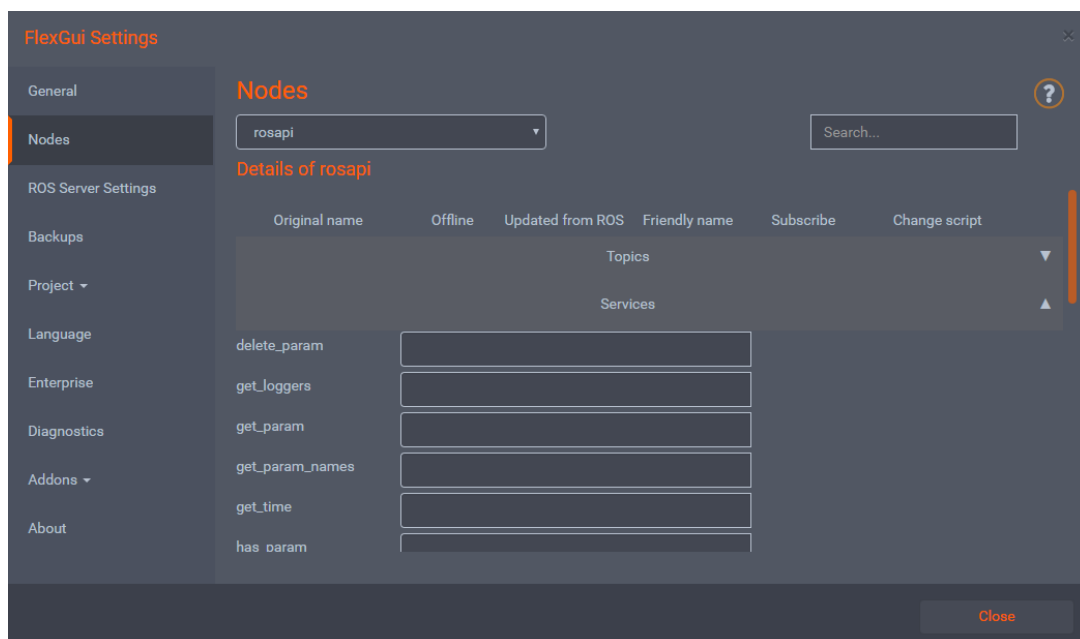
The init script editor is located under the project menu in the Settings window.

The init script will run after loading the project. It's used to initialize custom variables, functions. The init script is synchronized between devices, the value of the local variables after initializing them are not. If you want to synchronize values between two clients, please use server-side variables in ROS or a device.



3.2.3 Nodes

In this tab, the user can see the available topics and services in ROS, so it is not necessary to open ROS or node documentation to see what is available on each item.

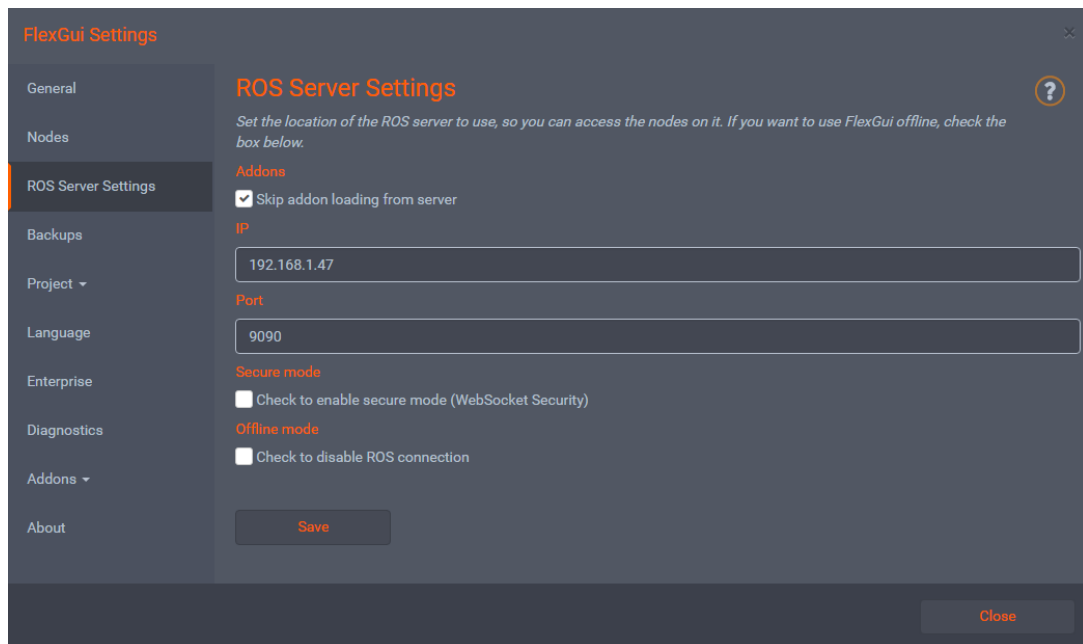


Both topics and services can be renamed for easier access, then the `@friendlyName` syntax is to be used instead of the full path. If a topic is subscribed, all changes will be received using push method, and all user interface elements will update automatically that use this value. Press Set button next to

the topic to create an OnChange script. This script will run after each topic update. When the edit button is visible, the topic has an already existing OnChange script and it is possible to update it. To remove the OnChange script, just delete the script from the editor.

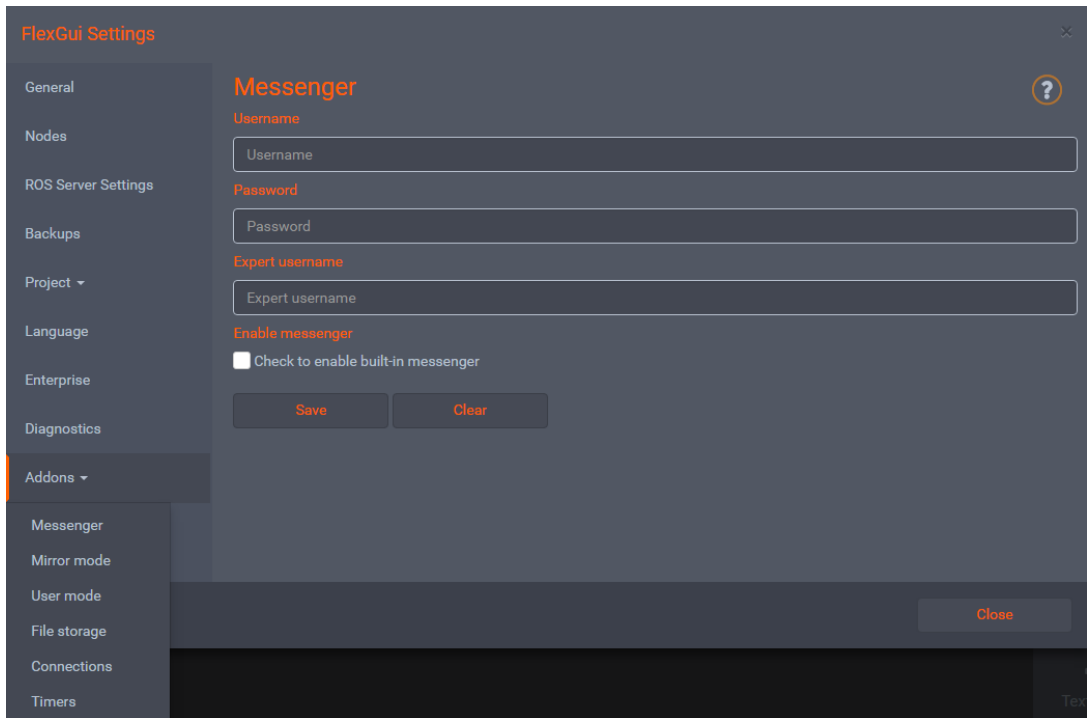
3.2.4 ROS Server Settings

In the ROS Server settings tab, the user can set up the address of the backend server or enable demo mode for local usage. The backend server should run roscore and rosbridge as well. In case you want secure communication, you need to enable WebSocket Security. This is required for the online trial as well. You can also disable the FlexGui Licensing server, so FlexGui won't use the licensed addons.



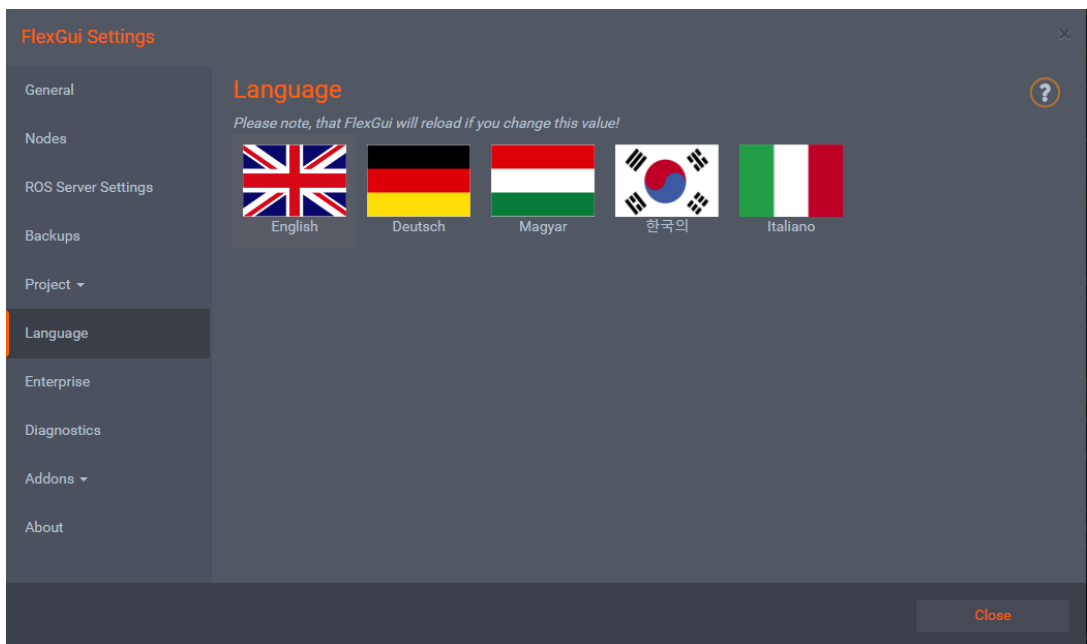
3.2.5 Messenger

FlexGui has a built-in messenger for remote support. To use this feature, the user has to set up the username and password before first use. FlexGui Messenger menu is under the Addons menu in the settings window.



3.2.6 Language

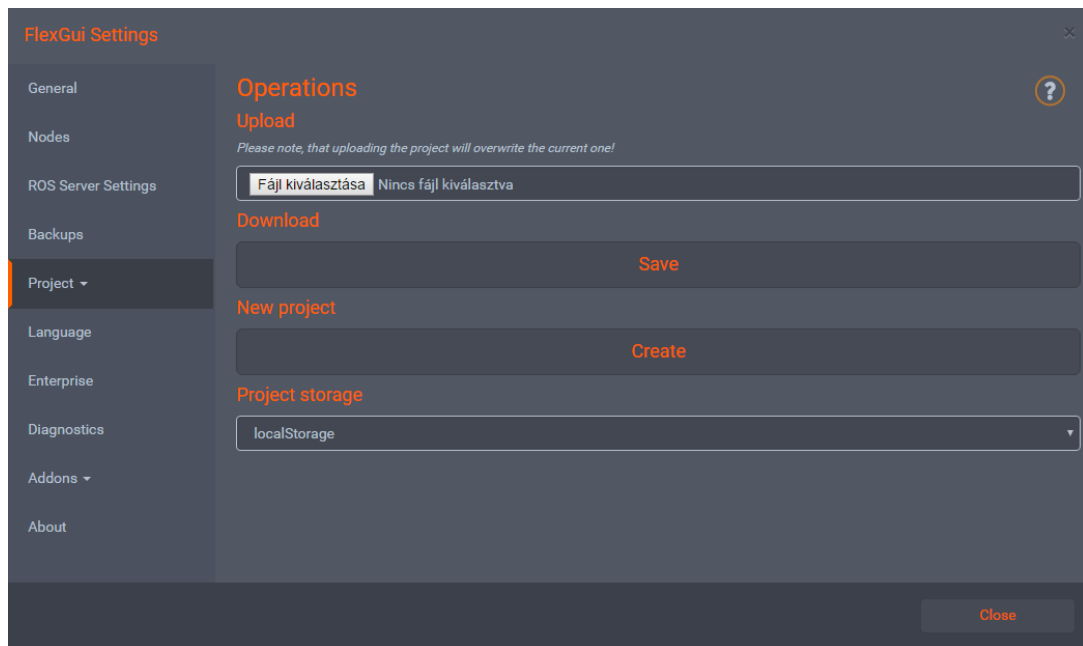
FlexGui supports multiple languages. The user can set up the currently used language in the Language tab.



3.2.7 Project operations

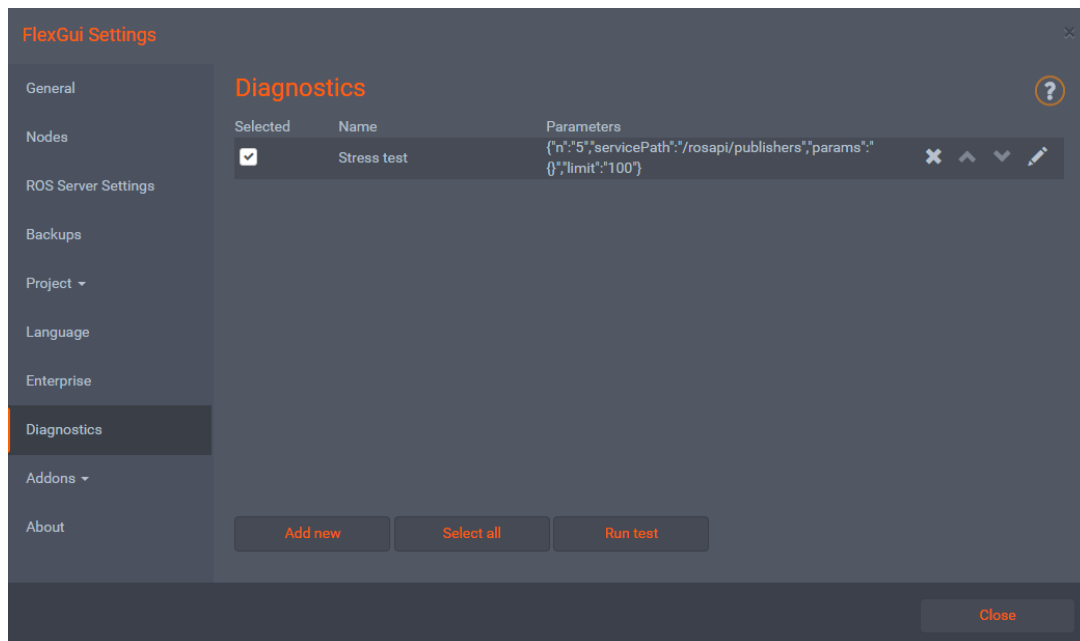
To upload and download projects, the user can use the operations tab under the Project menu. This makes it easy to backup or copy projects between devices, useful for debugging too. Demo mode also has this feature, so you can simply assemble a project in demo mode, then copy it to the working system or vica versa. If ROS is connected, the user can decide where to store the project:

- If more users are on the same ROS server, but want to use different projects, localStorage has to be chosen. With this, the project stays on the local computer and will not be synchronised to the ROS server.
- If more users want to use the same project, rosparam has to be chosen. With rosparam storage mode, the project is synchronised (manually or automatically) to the ROS server as a ROS parameter and each user will get the same project. Only one user can edit the project at once using the rosparam storage mode.



3.2.8 Diagnostics

With FlexGui 4.0 Diagnostics, the users can create test sequences to identify problems during runtime. Each test starts individually after each other and they are completely separated. The result of one is not affecting the others, thus with a good sequence, it is easy to find the problems in the nodes, connections or other parts of the system.



It is also possible to run only one specific part of the sequence, by removing / adding a test based on the selected checkbox on the left. The select all button will check all the checkboxes, thus all of the tests will be selected to run.

Pressing the Add new button opens a dialog window.

There are two types of tests:

Predefined tests can be selected from the drop-down list and each test has some predefined and required parameters (e.g. topic update rate's parameters are the nodeName, the timeout and the limit). Each parameter has a description, which helps the user to fill the form.

The time-based tests have a limit parameter, which supposed to target the maximum response time of a specific call.

User-defined tests can be written by checking the own script checkbox on the top.

New test

☒ Check this if you want to make an own script

Name

myTest

Script

```
1 /* put your script here */
```

Add script template

Save Close

Each user-defined test has to fullfill some criterias, to guarantee the proper operation. All of them can be found in the Help.

Please note: use the following variables for optimal result

- **test.result:** this will be shown on the UI under the name of your script
- **srv.testFinished();** Call this function when your script is finished, to be able to start the next test.
- **\$rootScope.\$apply();** call this function, when the script is finished, if you have any async calls (timeout, interval or e.g. waiting for a ROS event to end)

Pressing the Run button will run the selected tests. The result will be visible in the Result modal:

Diagnostics result

⌚ Stress test {"n":5,"servicePath":"/rosapi/publishers","params":{"limit":100}}

max: 49ms, min: 13ms, avg: 39.6ms

⌚ myTest

Everything is fine!

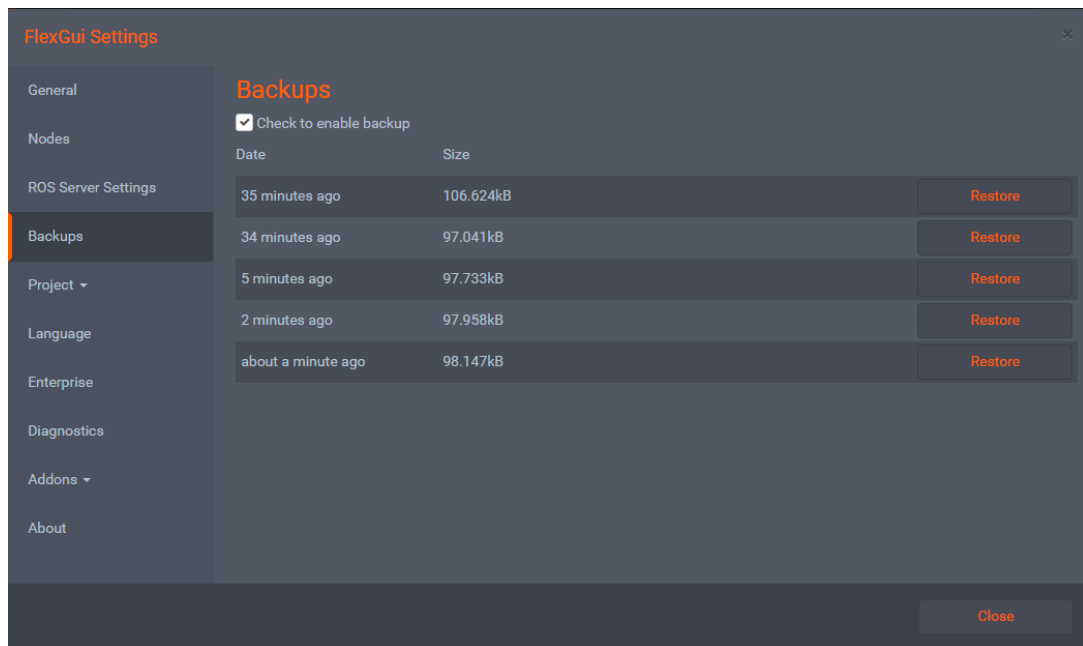
Cancel Close

The result clearly shows the outcome of each test. It is not possible to close the result window, until all the tests are finished, unless the user presses the Cancel button which stops the tests.

Please note: pressing the cancel button won't stop the current asynchronous call, but the next test case won't start.

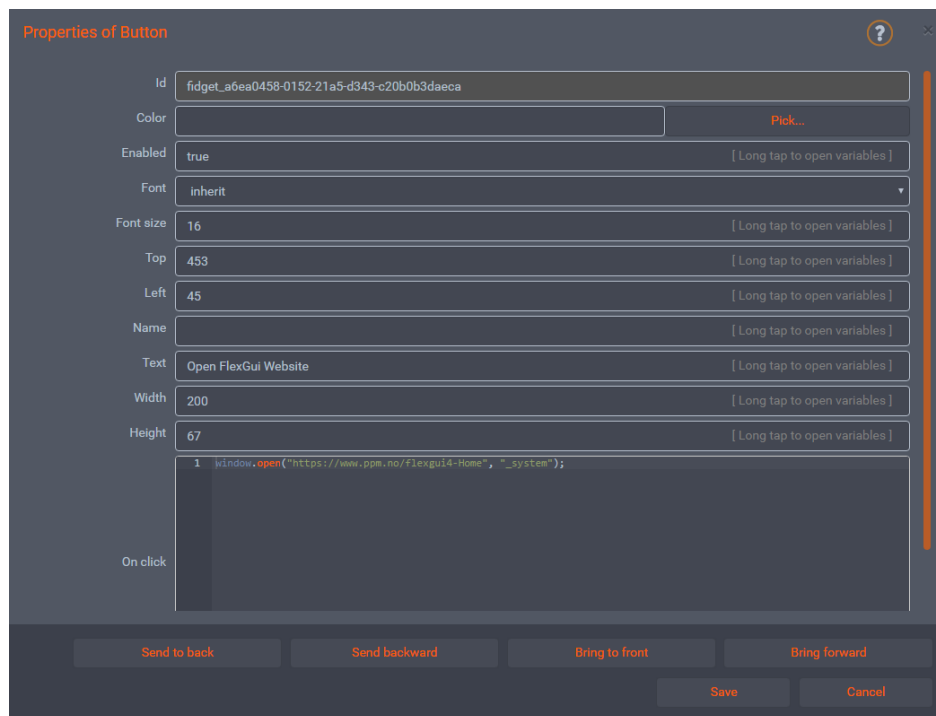
3.2.9 Backup

FlexGui 4.0 saves the current project to localStorage once every minute automatically, to backup the current work. In case of any problem, the user can go to Settings / Backup and choose from the last 5 backups.



3.3 Properties

To edit properties of screens or Fidgets, the user can use the Properties window. The content of the properties window depends on the edited item. The example shows the properties of a progress bar.



Arranging a Fidget (in z-order):

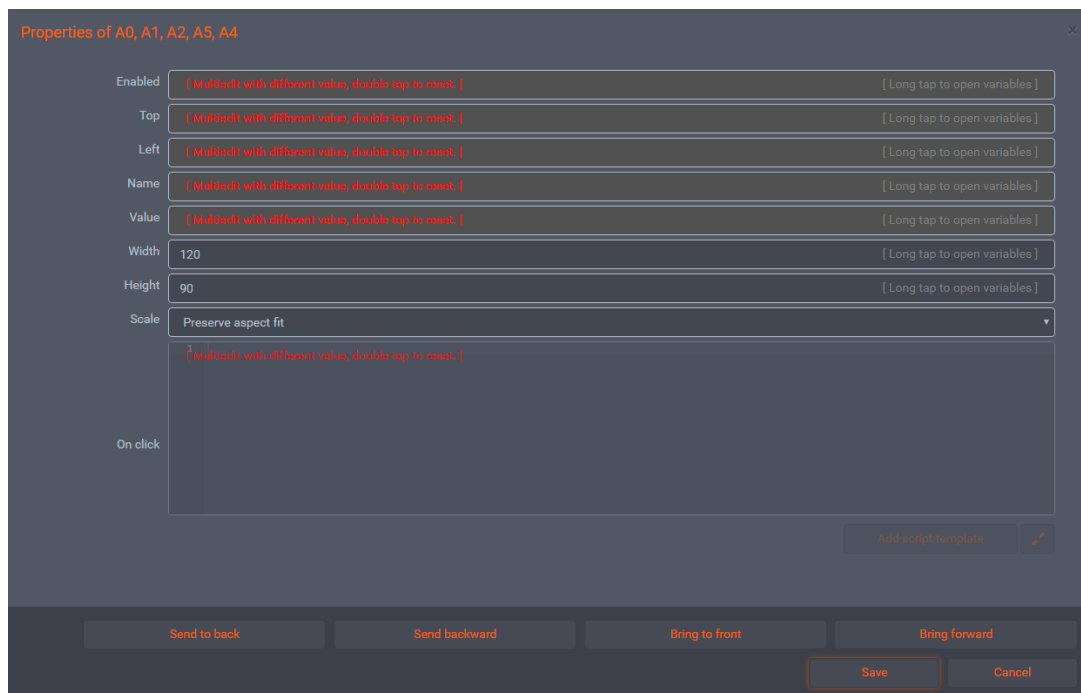


Ordering a Screen (moving it on the Screen Belt up and down):



Moving multiple Fidgets to the Properties Belt enables the multi property window. In this case, you will see the common properties of different Fidgets, such as top, left, width, height, name or etc.

If all of the edited Fidgets have the same value for a property, you can edit them easily. If you have different values, you have to double click the field to reset the value to the default one and edit all of them at once.



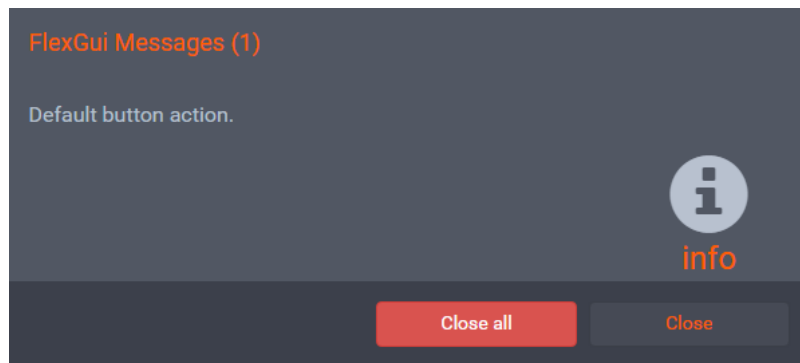
Saving the multi property window will overwrite all of the edited Fidgets' properties at once.

3.4 Popup messages

There are different types of popups in FlexGui:

- Info
- Warning
- Error

It is used e.g. if we cannot connect to the server. The user can show such screens programmatically using the #message call: `#message("some text", #infoMessage)`



3.5 Add screen

Defined earlier.

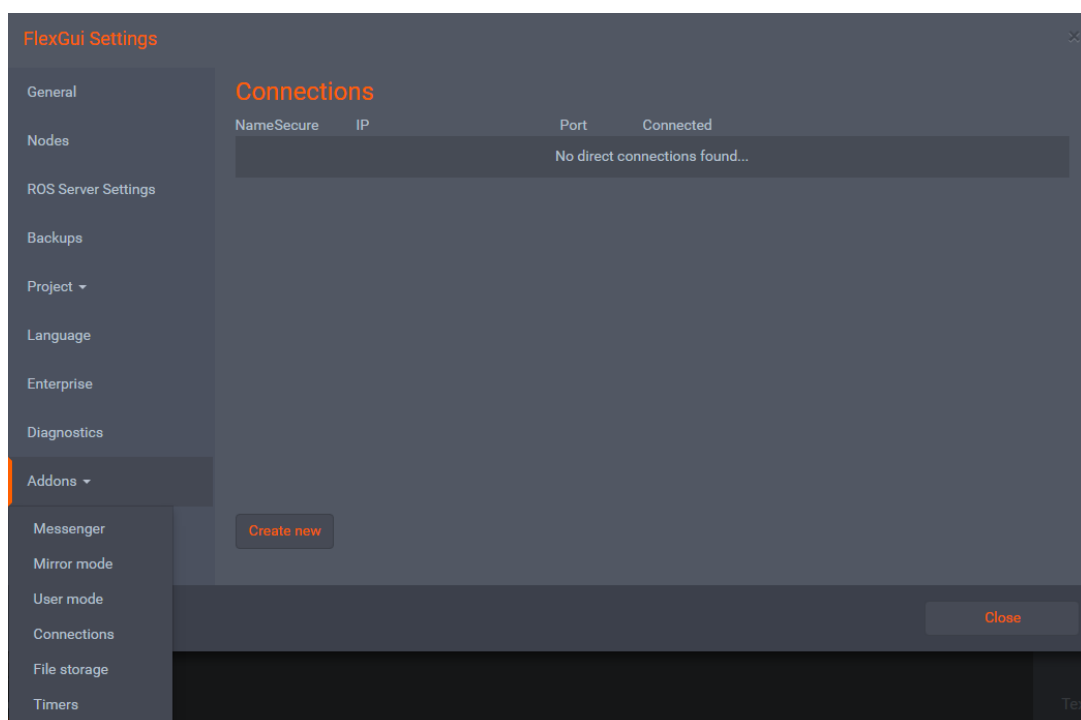
3.6 Login

Defined earlier.

3.7 Connections

In this tab, the user can create, edit and remove direct connections to the robot. The requirements of using the Direct connection functionality is to have FDLINK or FlexGui 2.4 license.

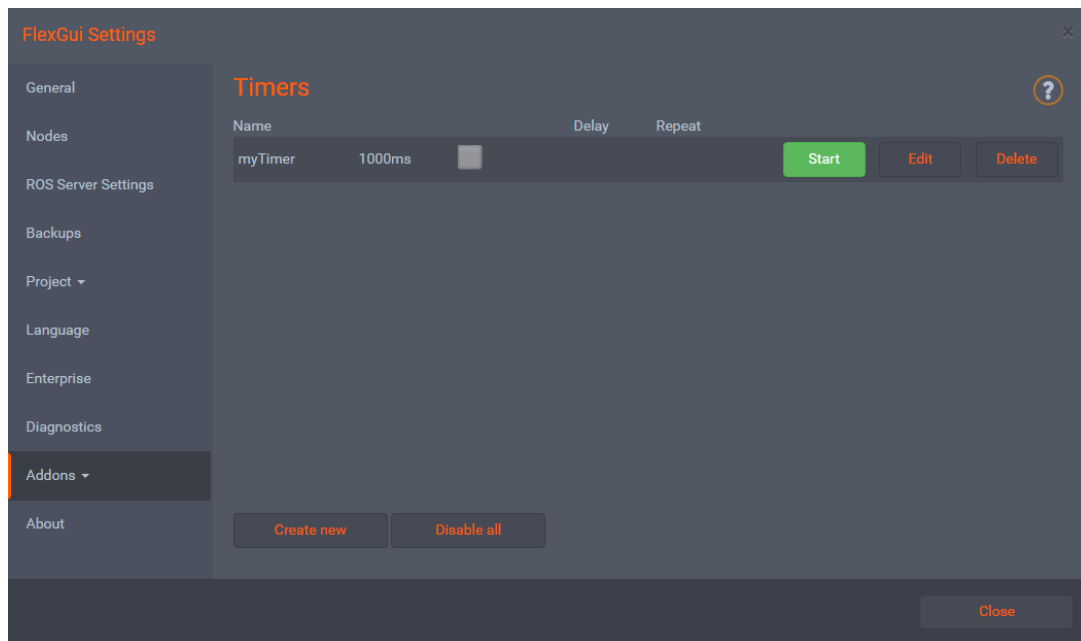
Clicking on the Create new button on the bottom opens the Direct connection editor window, where the user can set the parameters of the new (or if Edit was clicked, the existing) connection.



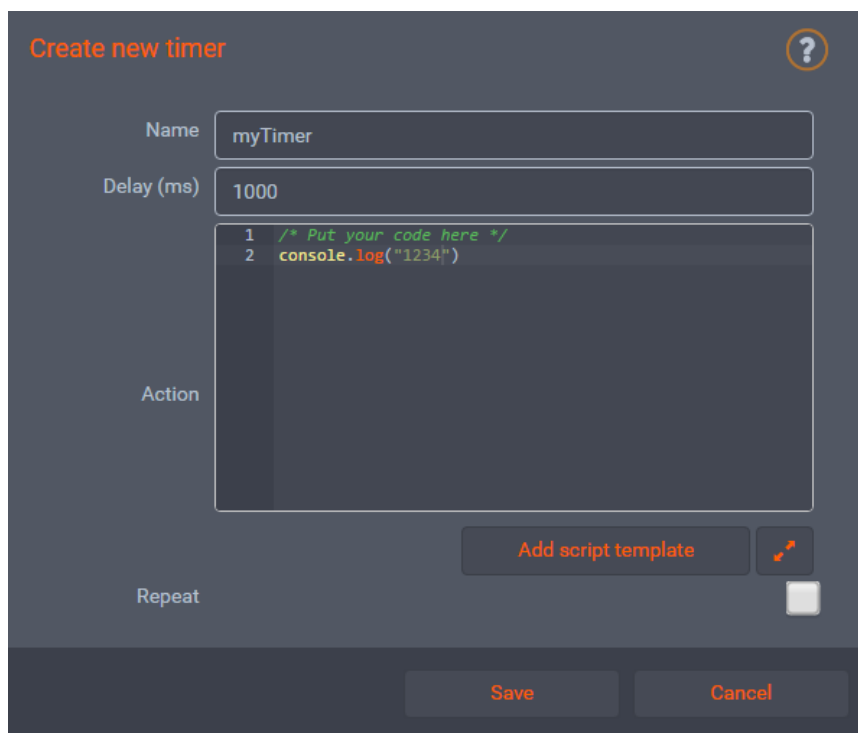
Working with Direct connections will be explained later in this document.

3.8 Timers

On the timers tab, the user can create and handle timers in FlexGui 4.0 in a centralized place. Timers are for making timed tasks in FlexGui 4.0. It is possible to create a timeout, to run an action after X seconds or create intervals, where the action will run in every X seconds repeatedly.



It is possible to start and stop a timer from JavaScript. For this, the user shouldn't use any special characters in the name, only numbers and letters for easier usage.



After creating a timer using the requirements above, the user can handle them like the following:

```
//this will start a timer, where the timer's name is "timer"
@timer.start();

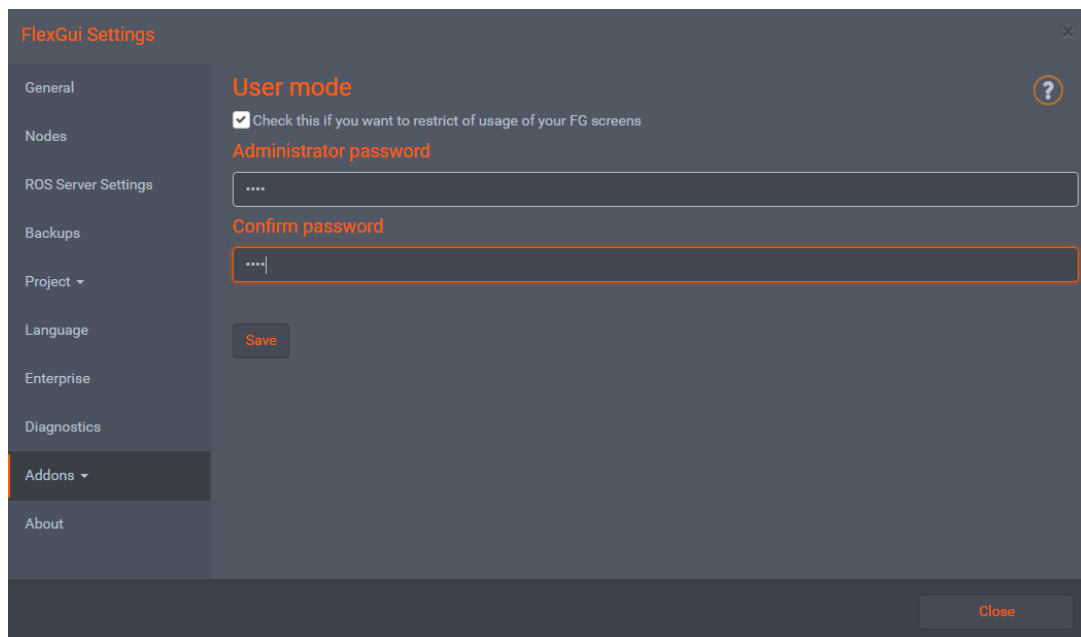
//this will stop a timer, where the timer's name is "timer"
@timer.stop();
```

If the user wants to use special characters, it is also possible to access to the timer:

```
//this will start a timer, where the timer's name is "My timer"  
variableService.friendlyCache["My timer"].start();
```

3.9 User Mode

FlexGui 4.0 can restrict the usage of FG Screens. This means, that it is possible to require authentication when trying to change the properties of a project, e.g.: entering to Edit mode, edit Screen properties, etc.



The password is stored encrypted on the server and only by that is it possible, to unlock the edit mode. To enable restricted mode, go to Settings / Addons / User mode, check the user mode checkbox, press Save and restart FlexGui. When **User mode** is enabled the user can see a Logout button in the Settings window. Pressing **Logout** hides the edit mode switch from the bottom right corner and shows the "Login" button. Pressing the Login button shows the login window, where the user can log in with the correct password.

Please note: The user name is "admin".

3.10 Enterprise

On enterprise tab, it is possible to try out most of the enterprise functions for free. To use this, please follow this step-by-step guide on our [website](#).

4 Moving Fidgets

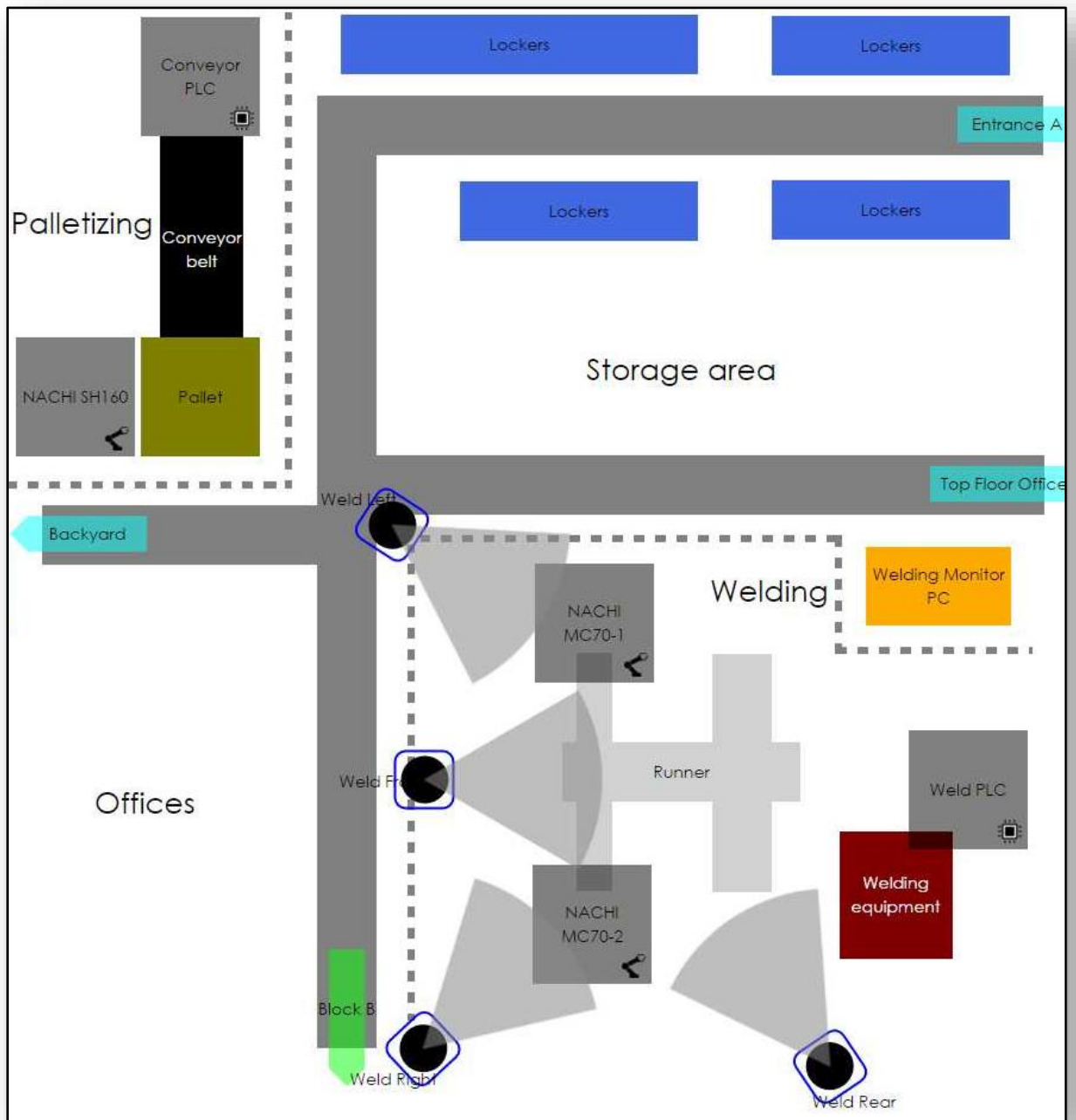
FlexGui uses drag & drop technology. This means, to move a Fidget, the user has to switch to edit mode, then grab it:

- From the Fidget Belt with a long press,
- From the screen with a drag or a click.

While the mouse is pressed or the screen is touched, the user can drag Fidgets on the screen. After the drag is released:

- opens the property window, if the Fidget is over the Properties Belt (the position won't update)
- deletes the Fidget, if the cursor is over the Delete Belt
- otherwise the Fidget will move to the current position on the screen

5 Factory designer



In the factory designer, the user can make a simple 2D overview of the installation without the usage of 3rd party image editors or any planning software. FlexGui also supports placing scripts on these factory Fidgets, so complex behaviour is possible to be implemented.

The factory editor mode has different Fidgets, as the normal mode:

- Fidget group,
- Text,
- Scrollable text,
- Road,

- Fence,
- Camera,
- Device,
- Obstacle,
- End of way.

See the details on the Fidget list (1.5.1).

The Camera and Device Fidgets also allow to specify a target screen, that is opened upon clicking them. This way the user doesn't need to write any JavaScript code.

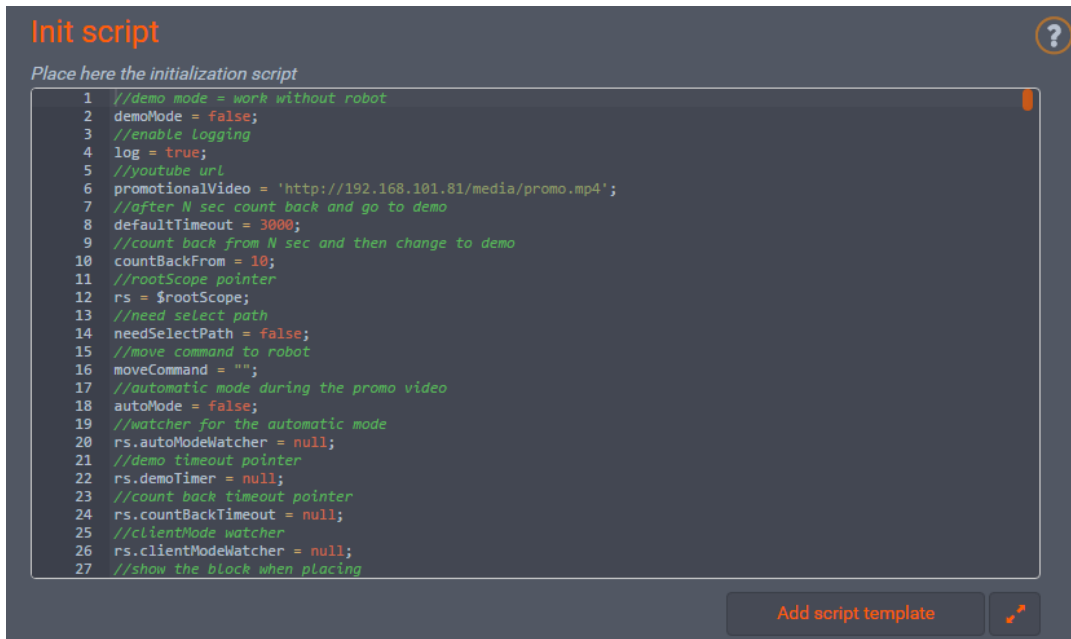
These elements can open a linked screen where the details of the selected object can be shown as debug information, camera image or any user screen.

6 Programming FlexGui

JavaScript is used in FlexGui to specify behaviour on specific events, like event handlers or button presses. The following section will explain the basic supported language elements through examples, for easier understanding of the system.

6.1 FlexGui Script Editor

FlexGui 4.0 with the Advanced scripting addon contains a JavaScript Editor, which supports code highlight, line numbering, automatic code indentation and full screen mode.




```
1 //demo mode = work without robot
2 demoMode = false;
3 //enable logging
4 log = true;
5 //youtube url
6 promotionalVideo = 'http://192.168.101.81/media/promo.mp4';
7 //after N sec count back and go to demo
8 defaultTimeout = 3000;
9 //count back from N sec and then change to demo
10 countBackFrom = 10;
11 //rootScope pointer
12 rs = $rootScope;
13 //need select path
14 needSelectPath = false;
15 //move command to robot
16 moveCommand = "";
17 //automatic mode during the promo video
18 autoMode = false;
19 //watcher for the automatic mode
20 rs.autoModeWatcher = null;
21 //demo timeout pointer
22 rs.demoTimer = null;
23 //count back timeout pointer
24 rs.countBackTimeout = null;
25 //clientMode watcher
26 rs.clientModeWatcher = null;
27 //show the block when placing
```

You can find this editor window everywhere, where complex coding is possible:

- Init script
- Onclick action editor
- Timers
- Diagnostics
- Topic onchange



Pressing the  icon makes the script editor fullscreen for better visibility, reading and editing. The **Add screen template** button is only visible, when FlexGui has the template wizards addon.

6.2 JavaScript function reference

FlexGui 4.0 contains some predefined functions, to help the users' work. All these functions start with #:

6.2.1 #autoInit

This function initializes a variable with a value, if it is not initialized before.

6.2.1.1 Parameters

name: string

value: object

6.2.1.2 Example:

```
#autoInit("int0", 42); //add int0 variable with 42, if it is not yet defined
```

6.2.2 #setScreenByName

This function changes the current screen to the given one using its name.

6.2.2.1 Parameters

screen: string

6.2.2.2 Example

```
#setScreenByName("New Screen")
```

6.2.3 #setScreenByIndex

This function changes the current screen to the given one using its name.

6.2.3.1 Parameters

screen: string

6.2.3.2 Example

```
#setScreenByIndex(1) //sets the screen to the 2nd on the Screen Belt, because the index starts from 0.
```

6.2.4 #message

This function pops up a message with the given string. It is possible to define the type of the message:

- #infoMessage pops up an info message
- #errorMessage pops up an error message
- #warningMessage pops up a warning message

6.2.4.1 Paramteres

text: string

6.2.4.2 Example

```
#message("Hello world"); //pops up "Hello world" az an info message
```

```
#message("Hello world", #errorMessage); //pops up an errorMessage.
```

6.2.5 @friendlyName

With this notation the user can access the ROS topics using a friendly name.

6.2.5.1 Example

```
@myTopic.value.data = 12; // set myTopic's value to 12, publishes the new value
```

Note: a variable is a class, that has several member variables as type, path, value. We use the value member, that is also a class. This, according to the documentation (http://docs.ros.org/api/std_msgs/html/msg/Int32.html) has a data member storing the value we need. Complex data types might have more member variables (e.g. http://docs.ros.org/api/geometry_msgs/html/msg/Pose.html).

6.3 JavaScript basics

6.3.1 Statements

JavaScript statements are separated by a semicolon (;). States are executed in the sequence they are written. Blocks in JavaScript are surrounded by curly brackets ({}). You can use blocks if you want to group statements.

6.3.1.1.1 Example

```
a = 5;
b = 6;
c = 12;
for (a = 5; a < 10; a++) //Explained later
{
    c = a + 2;
}
```

Please Note: JavaScript is case sensitive, variable **A** and **a** are different variables, while white space does not matter: **a = 5;** is the same as **a=5;**.

6.3.2 Comments

Comments can be placed in the code, just surround any text with `/*` and `*/`. If you write `//` anywhere in the code, the following characters are considered to be comments until the end of the line.

6.3.2.1.1 Example

```
a = 5; //This is a variable
/******
This is a comment that is
going through multiple lines
******/
b = 6;
```

6.3.3 Variables

Variables are present in JavaScript for storing data. Variable names can be letters, numbers and underscores (`_`), but they can't start with a number. The declaration creates the variable. It is possible to assign a value right at the declaration or after. The value of a variable can be modified infinite times. If you put quotes (simple or double) around the value of a variable it will be a string, if numbers are used only, it will be a number. If letters are used without quotes, it will be interpreted as a variable, throwing an error if it doesn't exist. Types of variables are dynamic, assigning text to a number variable will make it a string.

6.3.3.1.1 Example

```
//Declaring a single number variable:
var a;
//Declaration of several variables with or without value
var a = 5, b = "something", c, d = false;
c = a + 5;
a = b;
```

Please Note: redeclaration of a variable is possible, if there is no new value, the variable will remain exactly the same, as before.

6.3.4 Operators

Let's go through operators using an example. The comments next to statements show their results.

6.3.4.1.1 Arithmetic and assignment operators

```
//Arithmetic operators:
var a = 2, b = 3, c;
c = a + b; //c will be 5 (addition)
c = a - b; //c will be -1 (subtraction)
c = a * b; //c will be 6 (multiplication)
c = a / b; //c will be 0.666 (division)
c = a % b; //c will be 2 (modulus)
c++; //c will be 3 (incrementation)
c--; // c will be 2 (decrementation)
//Assignment operators:
c = 5; //c will be 5
c += 2; //c will be 7 (equals to c = c + 2)
c -= 3; //c will be 4 (equals to c = c - 3)
c *= 3; //c will be 12 (equals to c = c * 3)
c /= 4; //c will be 3 (equals to c = c / 4)
c %= 2; //c will be 1 (equals to c = c % 2)
a = "Hello", b = "FlexGui ", c = 3, d;
d = a + " " + b + c; //d will be "Hello FlexGui 3"
```

6.3.4.1.2 Comparison, logical and conditional operators

```
//Comparison operators:
a = 3, b;
b = a == 2; //b will be false (equal)
b = a === "3"; //b will be false (value and type equal too)
b = a != 2; //b will be true (not equal)
b = a !== "2"; //b will be true (not equal neither value or type)
b = a > 2; //b will be true (greater than)
b = a < 2; //b will be false (less than)
b = a >= 2; //b will be true (greater then or equal to)
b = a <= 2; //b will be false (less then or equal to)
//Logical operators:
a = true, b = false, c;
c = a && b; //c will be false (and)
c = a || b; //c will be true (or)
c = !a; //c will be false (not)
//Conditional operator:
//(condition) ? value1 : value2;
var a = 3 > 5 ? 1 : 2; //a will be 2
```

6.3.5 Conditional statements

Conditional statements can be used to determine, if a code should run or not depending on a logical statement.

6.3.5.1.1 Example of a conditional statement

```
if (4 > 6)
{
    var a = 5;
}
else
{
    var a = 6;
}
```

Because the result of $4 > 6$ is false, the else part will be executed, a will be 6. It is possible to use the statement without the else branch.

Please Note: curly brackets are used to group more statements, but not obligatory for one (if ($4 > 6$)
a = 5; is OK)

6.3.6 Loops

6.3.6.1 The for loop

The for loop executes the code block while the middle statement is true. The first statement is executed before the loop, the last one is executed after each loop.

6.3.6.1.1 Example of a for loop

```
for (var i = 0; i < 5; i++)
{
    //Statements
}
```

6.3.6.2 The while loop

The while loop executes the code block while the statement is true.

6.3.6.2.1 Example of a while loop

```
while (i < 5)
{
    //Statements
}
```

6.3.6.3 The do/while loop

The do/while loop executes the code block while the statement is true. The condition is checked after each run, so the body will be ran at least once.

6.3.6.3.1 Example of do/while loop

```
do
{
    //Statements
}
while (i < 5)
```

7 ROS-I connection

7.1 Overview

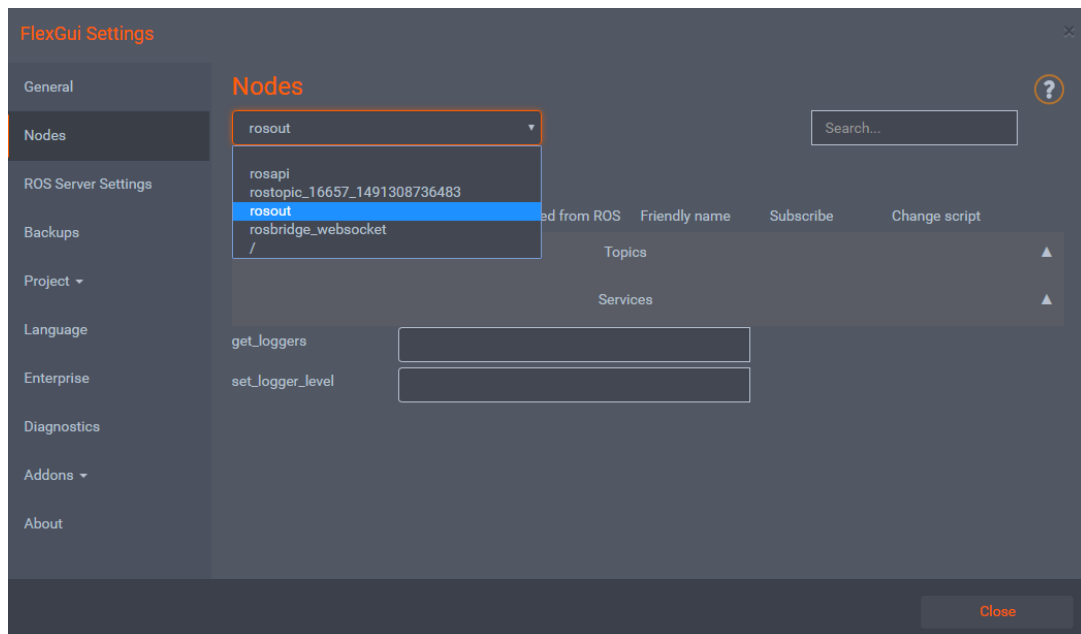
Supported devices

- NACHI robot connection,
- PLCs through modbus,
- KUKA implemented by Narvik
- ABB, Adept, Fanuc, Motoman, Universal Robot implemented by ROS

Additional features:

- Virtual environment simulation using Gazebo,
- One-click variable access
- Custom names to variables => short and clear code
- Normal PC or RaspberryPi both supported as ROS server

When the correct connection settings of the rosbridge server are set in the Connection settings tab, the available ROS nodes are shown in the Nodes tab.



Services and Topics are shown when selecting a node. The original names can be renamed and then accessed in FlexGui. In order to receive data from a Topic the subscription checkmark has to be set.

7.2 Publishing a Topic

Reading a topic is done using the `@topicName` syntax. To publish values, first you need to create and (if it doesn't exist yet) advertise the topic. This can be done in the `initScript` or for debug purposes in any control that has `onClick` script.

```
// PLC register topic creation
registerTopic = new ROSLIB.Topic({
  name: '/plc1/register1',
  messageType: 'std_msgs/Int32',
  ros: variableService.ros
});
// Advertise the new topic
registerTopic.advertise();
```

Then, to publish values, we need to use the topic we just created. This is preferably placed in an onClick script, but the initScript can have custom code like this as well.

```
//Publishing Int32 typed message
registerTopic.publish(new ROSLIB.Message({ data: 0 }));
```

Note: If you would publish another message type, you would need to use another syntax. Like a Pose has more members, a Point and a Quaternion.

7.3 Calling a Service

Without wizards you can reach the same functionality as the #callService wizard template, with some more typing:

```
//Creating the service object
var service = new ROSLIB.Service({
  ros : variableService.ros,
  name : '/rosapi/services',
  serviceType : 'rosapi/Services'
});

//Calling the service
service.callService(null, function(result) {
  //using the result
  #popup(JSON.stringify(result))
});
```

For more ROS - JavaScript samples please see the roslibjs documentation:

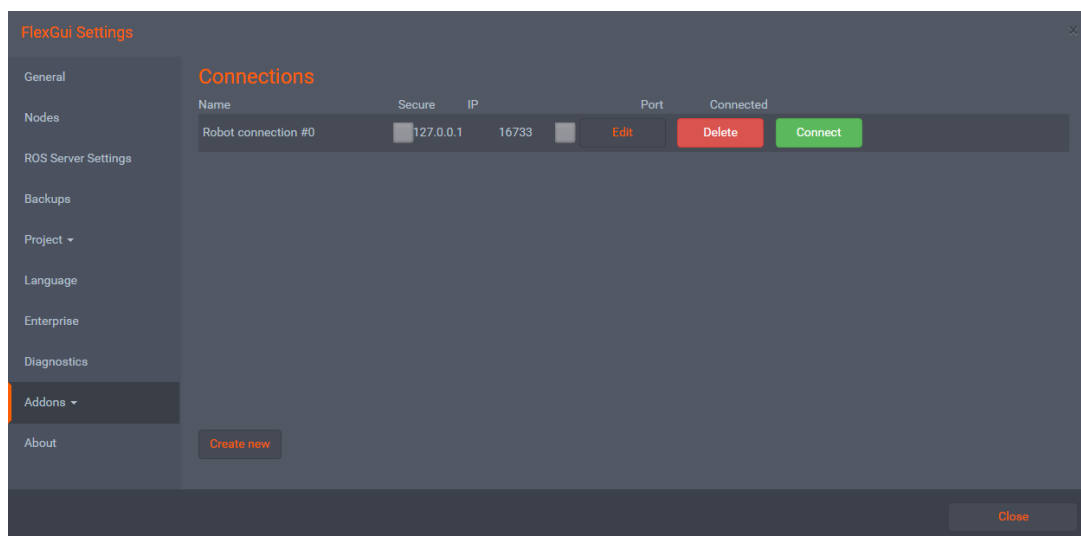
<http://wiki.ros.org/roslibjs/Tutorials/BasicRosFunctionality>

8 Direct connection

FlexGui 4.0 has the capability to create Direct connections (DC) with FlexGui-enabled robots.

Please note: To use FlexGui 4.0 in DC mode you will need FDLINK license. For more information please visit our [website](#).

To create, edit or remove DC, please go to Edit mode/Settings window/Addons/Connection tab.



To create a new DC, click on the Create new button on the bottom and fill the form. Please note, that the form is always prefilled, but the user can always change these values according to their hardware setup. After connecting, the Name will be changed to the FD's name.

The 'Create new' dialog form has four input fields: 'Name' (prefilled with 'Robot connection #0'), 'IP' (prefilled with '127.0.0.1'), 'Port' (prefilled with '16733'), and 'Secure' (a checkbox). At the bottom are 'Save' and 'Cancel' buttons.

To edit an existing DC, click on the Edit button.

To remove a DC, click on the Delete button.

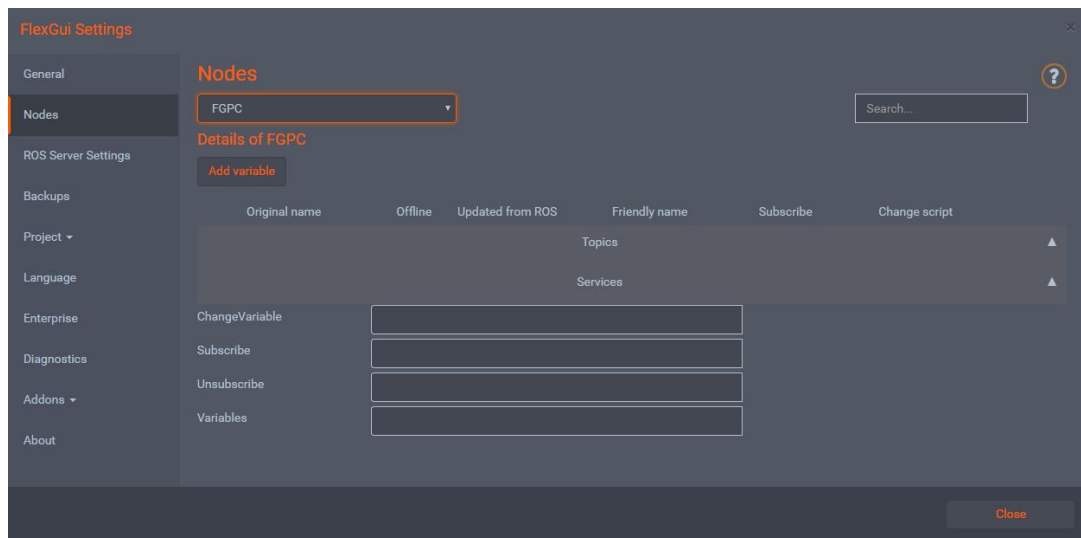
If the connection is UP, you will see a checkmark under the Connected column, next to the DC. If the connection failed or you want to reset the DC, you can press the Reconnect button.

After adding and editing a connection, FlexGui will try to connect. If it is successful, the user will see the checkmark under the Connected header. If the connection failed, an error will occur.

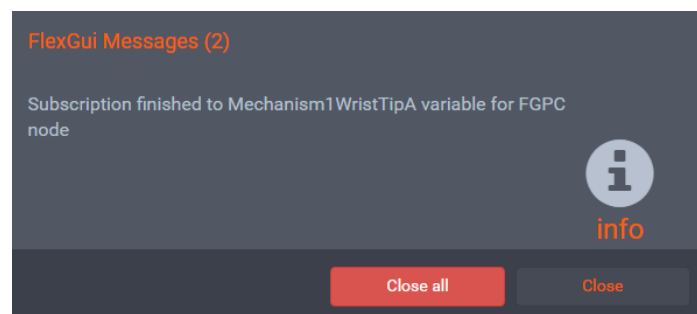
8.1 Variables and functions

The DC is working like the ROS-I Connections. If the DC has successfully connected to the robot, 4 new services (Variables, Subscribe, Unsubscribe and Change variable) will be added to the Nodes tab, under the Robot's node. To use it easier, on this kind of nodes, the users can use the Add variable functionality, just has to press the Add variable button, which appears on the top of the node's topics and services in Settings window/Nodes tab, after selecting a supported node. This will automatically show a list from the available variables and has the ability to easily (un)subscribe all of them.

Please note: The add variable functionality is also enabled through ROS for this kind of nodes!



After the subscription has succeed, the user will be notified.



8.1.1 Variables

This service will return all of the available variables from the robot.

Call

```
deviceService.nodes[<put your Node's name here>].Variables.call({},  
function(ret){ console.log(ret); });
```

Return

The list of variable names

8.1.2 Subscribe

This service will subscribe a selected variable on the robot. After the subscription, a new topic will be made and reported.

Call

```
deviceService.nodes[<put your Node's name  
here>].Subscribe.call({variable_name: '<put your variable's name here>'},  
function(ret){ console.log(ret); });
```

Return

This function doesn't have a return value.

8.1.3 Unsubscribe

This service will unsubscribe from a variable on the robot.

Call

```
deviceService.nodes[<put your Node's name  
here>].Unsubscribe.call({variable_name: '<put your variable's name here>'},  
function(ret){ console.log(ret); });
```

Return

This function doesn't have a return value.

8.1.4 Change value

Changes the value of a variable on the robot.

Call

```
deviceService.nodes[<put your Node's name  
here>].Unsubscribe.call({variable_name: '<put your variable's name here>',  
value: '<put your new value here>'}, function(ret){ console.log(ret); });
```

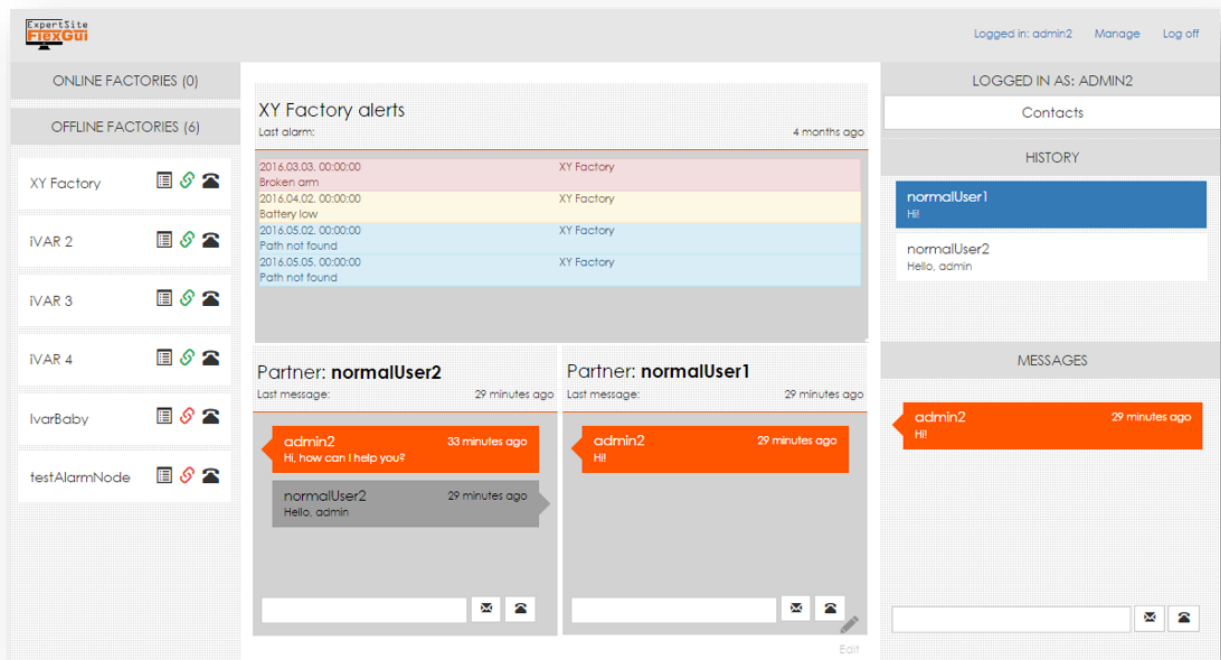
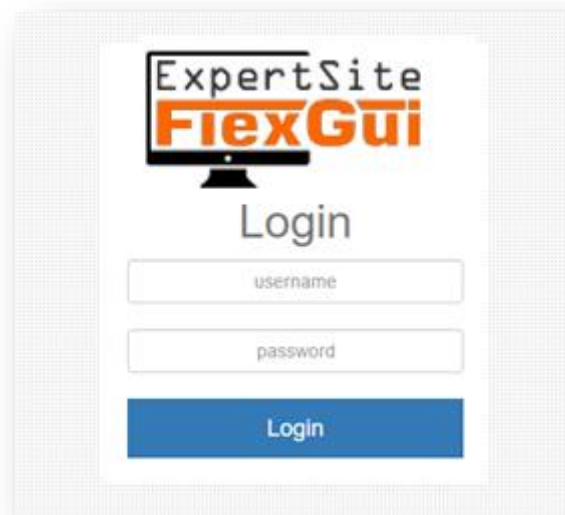
Please note: *FDLink accepts only string variable values, non-string types will cause an error.*

9 The ExpertSite

The ExpertSite is a tool to manage and control all available Factories, as well as a communication platform.

Expert Site uses a built in User Management System to hide Factories from different users. To log in, go to <https://expertsite.ppm.no> and use your account.

Only administrators can create new accounts.



The logged-in users can see all of their online and offline factories listed on the left side of the screen. This makes it simple for an expert, to manage more factories at once. Simple icons show the most

important information, through shortcuts, the expert can connect to a factory or call the local operator.

The Expert Site Fidget System is located in the center of the start screen. Pressing the small Pen icon in the bottom right corner will enable edit mode, where the user can add/edit the already placed Fidgets.

9.1 Expert Site Fidgets

There are 3 types of Expert Site Fidgets, which the user can place on the main screen, the dashboard.

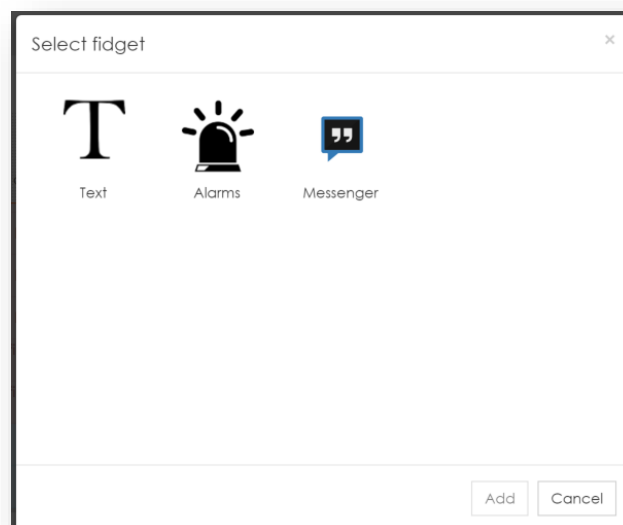
With the alarm Fidget, the user can monitor and filter the latest alarms. To place an alarm Fidget, press the small pen icon in the bottom-right corner. Using this, the user can enter to edit mode, and will see the Edit Belt on the top.



Here the user can (from left, to right):

- add a new Fidget
- enable the snap to grid functionality, to help aligning Fidgets
- set the properties of a Fidget
- copy
- cut
- paste
- undo
- redo
- delete

Pressing the add new Fidget icon will show the Fidget selector window.



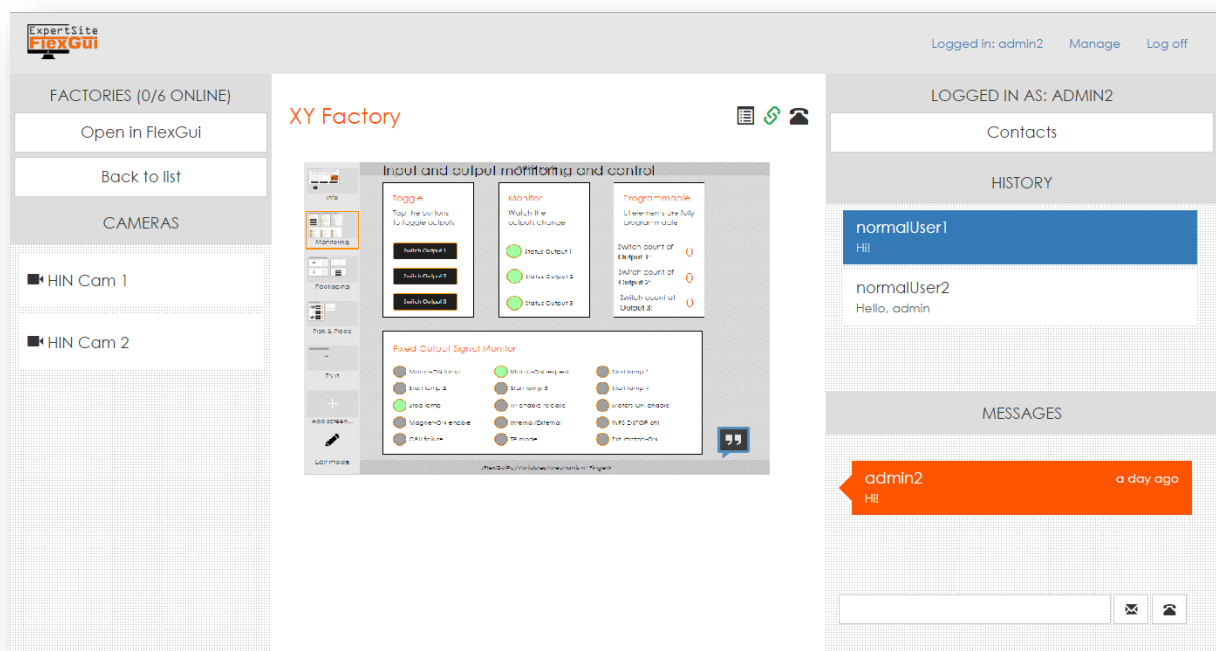
With the **Text Fidget**, the user can add customized texts on the screen.

The **Alarm Fidget** allows the user to show the most important alarms on the main screen. For this, the user can filter the alarms by severity, factory and count. If an alarm Fidget is placed on the dashboard and the Expert Site is open, a sound will be played, when a new alarm appears in the list. The alarm Fidget has 3 different layouts and if an item is clicked, it will show a detailed window with the message of the selected alarm.

The **Messenger Fidget** is used to place conversations on the dashboard with a selected user.

The user can place as many Fidgets as he wants and where he wants. Thus, it is possible to place multiple conversations with different users on the dashboard as well as multiple alarm Fidgets with different filtering.

9.2 Factories



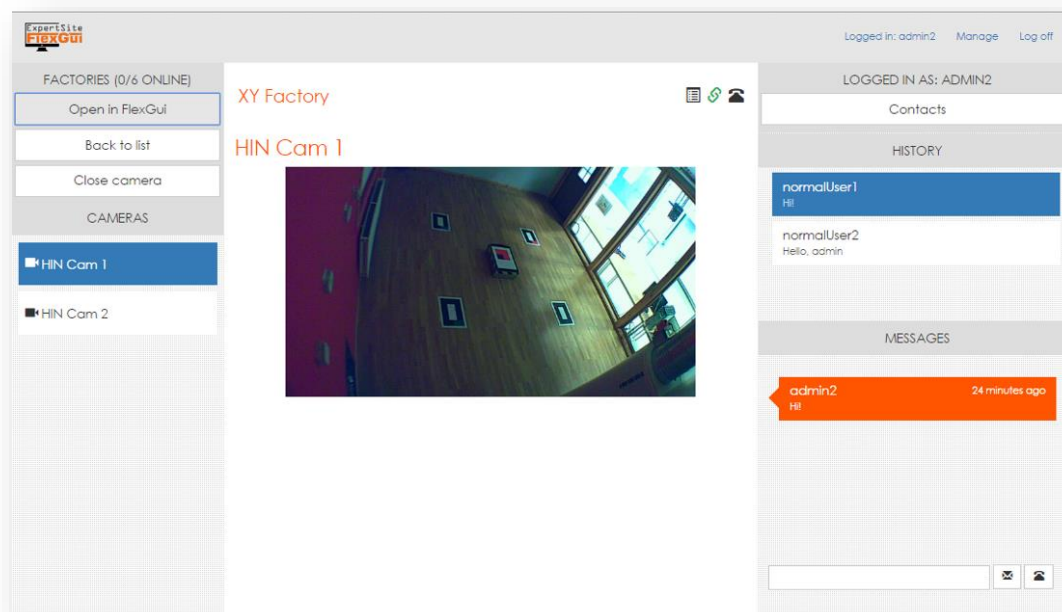
When selecting an online Factory, the screen mirror of the Factory is visible, allowing the expert to see the same FlexGui screen, as the Factory user (mirror mode has to be activated in FlexGui). Available cameras are shown and can be accessed.



When the expert clicks into the FlexGui window the cursor appears at this point. This cursor is transmitted to the FlexGui interface of the local user.

The cameras and the Open FlexGui functionality require VPN (or direct) Connection to the Factory's network. Open FlexGui button will open a new window, redirects the user to FlexGui Trial using the set up ROS Connection Address.




Clicking an item on the Camera list will open the Camera page and shows the selected camera's image.



It is also possible, to open a FlexGui and see the cameras' image in a FlexGui 4.0.

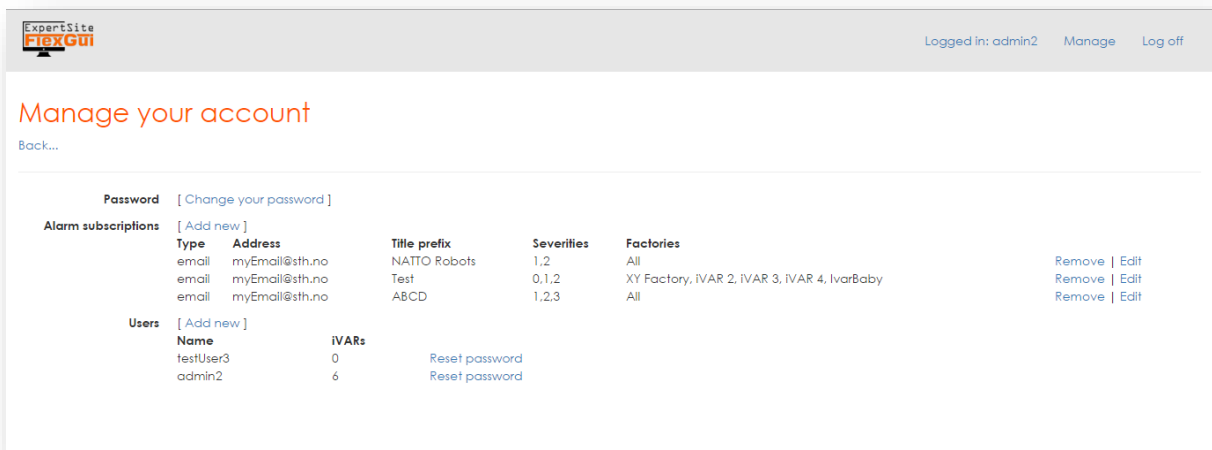


Symbol explanation of the top bar:

	Alarm list for the factory
	Connect to the factory's FlexGui (VPN required)
	start a video call with the factory user

9.3 Manage screen

The logged-in users can change their preferences in the Manage screen. Pressing the Manage text on the top of the window will redirect to the Manage screen.



The user can change his password and create **Alarm subscriptions** here. There are 3 different types for an alarm:

- **Email:** if an alarm fires, an email will be sent to the selected email address
- **SMS:** (only on customer request)
- **Push:** if an alarm fires, the registered devices will show a new notification in the notification bar. To register your device, go to the Manage screen on your device's Chrome browser and follow the instructions. If you want to remove a push device, go to the manage screen and press the Remove link next to it.

Push devices	Name	Added
	Device #0	8/26/2016 10:06:16 AM
	Device #1	8/26/2016 10:08:18 AM
	Device #2	9/9/2016 12:42:19 PM
	Device #3	9/14/2016 1:24:18 PM
	Device #4	10/5/2016 12:52:58 PM

9.3.1 User management

Company users can create new users and assign his factories to different users, to balance the management between users.

Please note: only the administrator can create Company users, please [contact us](#) if you need it.

If you are logged in with company users, in the Factory – User matrix you can assign a factory to a user.

My Factories	Name	c1user3	normalUser4	normalUser1	normalUser2
	testAlarmNode	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	testAlarmNode-2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	fabianAlarmNode	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	enricoAlarmNode	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	TestNode	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

If a checkbox is checked, the selected user will be able to use the selected factory. Otherwise the factory will be unavailable for the user.

All users can see the available factories on the Manage screen, next to the **Available factories** heading.

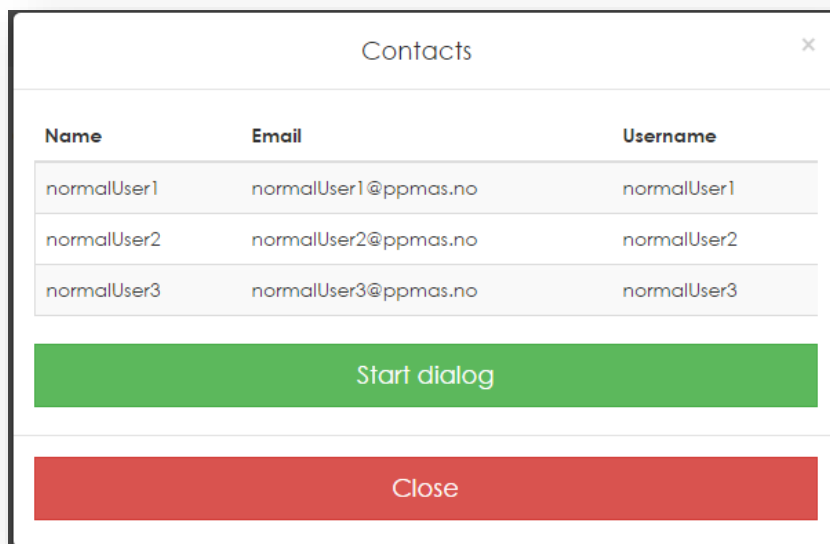
9.4 Messenger

A chat dialog for communicating with a local user can be found on the right-hand side. The expert can communicate using video, voice, text. On the FlexGui side, there is a single button to call the expert and also a Chat Fidget to exchange messages.

The built-in messenger is using QuickBlox engine, a 3rd party messaging solution, specially for mobile and browser use.

9.4.1 Contacts

Each user can create his own contact list. Pressing the Contacts button on a logged in messenger will open a new modal window.



This window contains all the available contacts for a specific Messenger account.

9.4.1.1 Setup example

Expert user's account: expertUser / expertPassword

Expert user has 3 iVARs, so the user creates 3 contacts in the Contact window / Create user tab:

- ivarUser1 / ivarPassword
- ivarUser2 / ivarPassword
- ivarUser3 / ivarPassword

Now the expert user can set up the messenger on the iVAR's in FlexGui 4.0, by using the Settings window / Messenger tab:

After saving the form, FlexGUI will reload and use the new settings for the messenger. Pressing the Quick Call button in FlexGUI 4.0 will call the expertUser, and the expertUser will be notified, that he has a caller: ivarUser1.

9.5 Alarm subscriptions

FlexGui ExpertSite makes possible to receive alarms from FlexGui and forward them in email / push / sms message.

To create alarm-forwarding rules, go to Manage, where you can find the Alarm subscription. Here you can add, edit and remove alarm rules.

This will be in the message's title: [<myPrefix>] Alarm title

Severities

Forward the alarm if the severity is in the list. If the list is empty, forward all.

Notification address

The address, where the alarm will be forwarded.

Type

The type of the address (email / sms / push)

Factories

List of the selected factories. If empty, forward all.

9.6 Accounts

To use all the functions of the Expert Site, the user needs an account:

- Expert Site account: to access the site, check iVAR states, use mirror mode, etc.

This account can be requested in email: admin@ppmas.no.

10 FlexGui Node

FlexGui Node (FGN) is a low-level ROS node that supports FlexGui in operations that require high speed or constant running.

To run the FlexGui Node the FlexGuiNode executable file needs to be started.

Configuration and setup

Debug mode

More information can be printed to the output of the inner-workings of FGN by using the debug command line parameter upon startup.

```
./FlexGuiNode debug
```

This additional information is saved as a log file, all the time, under the file name: log.txt. Note, that using debug mode might slow down the operation cycles.

Configuration file (.../Testversion/PC/flexGuiNode.ini)

The FGN is a host for other subcomponents, that are running independently from each other. FGN is therefore only responsible to keep the connection to the ROSbridge. The only configuration option of FGN is the address of the ROSbridge. Use the following format to set up:

```
rosBridgeAddress=ws://localhost:12345
```

To indicate the use of secure websockets, use wss as protocol in the address.

10.1 High Speed Camera Proxy

FlexGui Node supports acting as a proxy, to avoid entering username and password when connecting to cameras. After logging in to the camera, the received data is simply forwarded through the proxy, no processing is done to conserve resources.

Configuration and setup

The list of cameras used by the Camera Proxy is set up using the cameras.ini configuration file. A camera setup looks like the following:

```
[Example Camera]
url=http://myurl.com/video/mjpg.cgi?profileid=3
port=12345
user=exampleUser
pass=examplePass
```

To add more cameras, simply copy these lines under each other and modify according to your needs. If a camera is not used for a while, it is possible to comment it out by placing semicolons before it's name like this:

```
;[Example Camera]
```

The **url** must point to the mjpeg stream of the camera. Tip: If the address is copied to a browsers' address bar, the video stream should start playing after logging in.

The **port** tells the Camera Proxy on which port should it host the camera. The ports of different cameras should be different. If there is already a service registered to use this port, the camera will not work, as only the first registration is accepted to a listening port.

The **user** and **pass** are used to log into the camera, so in FlexGui the end user doesn't need to do it with all devices at all startups.

Interface

To access the Camera Proxy from FlexGui, a ROS interface is implemented. Calling the `get_cameras` service will return the list of available cameras with their full url in a string list. This way it's possible to iterate through all cameras and dynamically handle changes.

An example call of this service:

```
rosservice call /FlexGui/get_cameras
```

10.2 Alarms

As FlexGui Node is always running, it can watch for specific changes happening with topic values. The Alarms component subscribes to a list of topics, and calls a JavaScript script when a new value is received. The script can return simple or complex information about the data received.

Configuration and setup

Configuration file

A typical configuration file looks like the following:

```
;Enter topic names separated by space (" ")
watchedTopics=/FlexGui/Variables/Output1000 /FlexGui/Variables/Output1001
/FlexGui/Variables/SafetyPlug /FlexGui/Variables/MotorsOnLamp
/FlexGui/Variables/InPlaybackMode /FlexGui/Variables/ESTop
expertSiteAddress=https://expertsite.myurl.com
expertSiteUserName=demoUser
expertSitePassword=demoPassword
nodeName=myAlarmNode
reportTimeout=5000
```

To comment out a line the user can place a semicolon in the beginning of the line. This helps to store more configurations, without deleting them completely.

To set up the **list of topics** to watch, simply enter their full path separated by a space. Configuration changes are loaded at startup, not refreshed during runtime.

Alarms are sent to the ExpertSite, so it's required to set up the **address** of it in the configuration file. This setting should be the base address where the login page appears in a browser. The Alarms component will log into the site and set up communication channels.

The **user name and password** are used to tell the ExpertSite, under which user's account the data should be stored. Other users won't be able to see these reports.

The **node name** specifies the source of the alarm on the ExpertSite, if more reporting agents are present in the system, this will help to identify the source.

The **reportTimeout** is the delay between two reports sent. Even if many alarms are received during two reports, they are collected to a cache and sent in one package to save resources. Be aware, that setting this time very low might slow down the whole system.

Alarm script

When a topic update is received, the `alarms.js` file is parsed and executed every time. The file this way is read on every update, so the script can be changed during runtime without restarting the FlexGui Node.

A typical configuration file looks like the following:

```

//Provided variables are:
// string topic: the name of the topic
// Object data: the object sent as topic message
//Return format is {severity: 1, message: "default"}
//If the returned value is null, it will be ignored.

if (topic == "/FlexGui/Variables/Output1000") {
    if (data.value == 1)
        return { severity: 1, message: "The robot was moved left." };
    else
        return null;
}
else
    return { severity: 3, message: "Not specified: " + topic };

```

This file is using JavaScript syntax, that means, basic JavaScript knowledge is required to make alarms.

The script is provided two variables: one is the name of the topic as string, the other is the data received in the topic as an Object. The type of the data is depending on the topic type, the user has to be aware of it. In the example the data has a field called value that we use to decide if we need to send an alarm.

The scenario is the following: we are watching for changes of a topic. If any other topic update is received, we send an alarm about a not specified topic that we can't handle. If the correct topic is received, and the value is 1, so the Output1000 is ON, then we send a report using severity 1, and an informational message. Messages can be assembled using the received data:

```

return {
    severity: 2,
    message: "Motors were turned " + (data.value == "1" ? "on" : "off")
};

```

This way it's simple to give some more details about the alarm to the ExpertSite.