**The iRidium software package** is developed for control of different automation systems from iOS/Windows/Android/OS X devices.

iRidium is a product meant for use of professional installers and integrators of automation systems, audio-video systems, security systems, etc.

To create the visualization project in iRidium:

## 1. Register on the iRidium Mobile web site.

Registration >>
You will be able to download iRidium products and make purchases on the web site (after Full registration).

## 2. Install iRidium Environment on PC (Windows 7/8).

Download >>
iRidium visulalization interfaces are developed on PC.

## 3. Install iRidium on your control panel (iOS/Windows/Android/OS X).

Install >>
You will be able to upload your visualization project on the control panel with i2 Control.

## 4. Decide what equipment you want to control.

See all the products >>
It helps you to find needed documentation.

## 5. Take training courses in iRidium Academy

**1.** Introductory iRidium webinar (online or video) – how to start work with the iRidium software package
**2.** Work with iRidium GUI Editor - for creating visualization interfaces (online or video)
**3.** Control of Audio-Video equipment (online)
... and other free webinars or video lessons.

## 6. Select a Sample GUI for your project.

see ready GUIs >>

- Create an interface with the help of ready Gallery in the editor
- Use wizard interfaces
- Sample GUIs from the web site (buy or download for free from the web site)
- Create your own custom design

### 7. Select the modules for controlling AV equipment.

See ready modules >>
For example: Sonos, iTunes, XBMC, Squeezebox, Global Cache, Kramer, Russound, Samsung Smart TV, Yahoo weather widget, clock, calendar, screensaver ...

### 8. Get Trial licenses

Trial licenses work with any iRidium drivers and scripts. The only limitation: 30 day validity period.

### 9. Upload the ready project on your control panel

How to upload projects >>

↑ Back

## Terms and Definitions

Main terms and definitions

- **iRidium Project** – a project file created in iRidium GUI Editor. It consists of a graphic part (GUI) and a driver part (commands for controlling equipment). iRidium Projects set up in GUI Editor have *.IRP or *.IRPZ formats. Projects consisting of the interface and driver parts is processed by the iRidium Transfer application and uploaded on control panels with installed iRidium Apps.
- **Graphic interface** – the complex of items of iRidium project graphic part (project pages and popups, graphic items).
- **Page /popup** – objects which define the interface structure. They are meant for placing graphic items. You can open only one project page at a time. It is a project base over which you can place any number of popups of different size.
- **Graphic item** – an object on a project page or popup (button, level, animated item, inertial list, etc.)
- **Control panel**– an iOS, Windows, Android, OS X based device with the installed iRidium App where it is required to launch iRidium project.

- **License** (license file) – an *.irl format file which enables connection between a panel and

controlled equipment. The license is uploaded onto the panel together with the iRidium project.

- **Activation key** – a set of 160 symbols which serves as an iRidium license activation code. It contains information about the acquired license and enables you to generate a license file.
- **Licensing#Activation of iRidium Licenses|Process of key activation]]** – creation of a license file with the help of the key in the MY LICENSES section of the web site.
- **Control panel identifier (HWID/UDID)** – the unique identifier of the Apple iOS (UDID) or Windows (HWID) controlled device, which is required for iRidium key activation and license file receipt.

- **Emulator** – a tool for simulation of the project launch on the control panel. It enables you to see how the project works and check communication with the equipment.

- **Driver** – a description of a protocol for equipment operation created in iRidium. It allows iRidium projects to refer to the equipment of the corresponding type and receive data from it. iRidium supports a number of native (written preliminarily and stored in iRidium Apps) drivers. There is also a possibility to create custom drivers on the basis of AV & Custom Systems with the help of IRidium DDK – it enables profound configuration of the driver with the help of the script system.
- **IRidium DDK** – a tool for creating drivers. It includes iRidium Script machine, iRidium Script API, iRidium DDK for writing drivers and examples of drivers.
- **Controlled device** – equipment (a controller, transmitter, media server, AV device, etc.) which uses one of the drivers described either in native or script part of iRidium project.
- **Command** – an instruction for sending data to controlled equipment. The command sending is activated by pressing on a graphic item, system events or scripts. It contains information about the data type and settings specific for particular equipment. The list of instructions (commands) is stored in the Project Device Panel tree.
- **Status channels (Feedback)** – an instruction for processing data received from the controlled equipment. Received data can affect GUI items (value output on the screen, switching item states). Status channels are stored in the Project Device Panel tree.
- **[[Project Token** – a variable for saving data received from control equipment or interface. Project Tokens serve for writing, storing data or transferring them inside the project. The data can be numerical (in the DEC format) or string (in the ASCII format). After the restart of iRidium Apps or control panels information written in Project tokens stays available (for example you can write a login and password for getting access to the equipment, connection properties – IP-address, device password, etc).
- **System Tokens** – variables storing data about the control panel state (time, date, data from sensors, etc.). System tokens cannot be affected via interfaces, they can be read only.
- **Driver Tokens** – variables storing data about the state of connection to project drivers and connection properties. Driver tokens cannot be affected via interfaces, they can be read only.