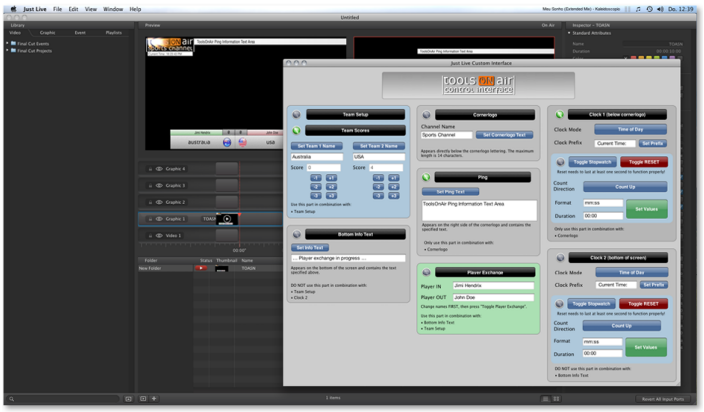
Custom Graphical Control UI for the FX Layer

In just:live and just:play it is possible to use a **custom graphical control user interface** for the FX Layer. Instead of using the Inspector to change parameters of a real-time graphic (composition:builder or Quartz Composer file), a dedicated graphical user interface is used to control the graphic.

> In **composition:builder** and **Quartz Composer** you can publish **Input Ports** to make them accessible by the playout, which means that the user can modify the published Inputs in the Inspector of just:live and just:play. For many cases this is an easy and sufficient way to set, control and modify the content and behavior of real-time graphics. But there are situations where Input Ports have to be controlled in a more defined and more structured manner.



Features:

- Creation of custom graphical control user interfaces for sports, news and other purposes.
- Control interface can be used directly in just:live/just:play (as a floating window) or in a web browser.
- Usage on mobile devices like tablets, fablets and smartphones.
- Platform-independent, i.e. can be used with any HTML-conform web browser.
- HTML based custom UI can be created in any HTML-conform text editor or website development environment.
- JavaScript, CSS, XML, CGI and PHP can be incorporated to create dynamic interfaces (local webserver may be required).

> Communication of a custom control interface can only take place in an **unidirectional** manner, that means that the UI can send values to a real-time graphic, but it cannot "receive" anything from the playout system.

Requirements

- A **separate** system with a working just:live or just:play channel. Channel activation is **not** required, as no actual playout can be done on this particular system.
- composition:builder and/or Quartz Composer, installed on the same system.
- A HTML-conform text editor or website development environment.
- Service USB Driver and TOAJustOutVideoRenderer Frameworks.
- Disabled TOA System Checker.
- TOA Quartz Composer PlugIns.

Preliminaries

Installing the Frameworks & Plugins

> Close **all** applications on your Mac before installation.

- Put the **Service USB Driver** Framework into the folder:
  - **/Library/Frameworks**
- Put the **TOAJustOutVideoRenderer** Framework into the folder:
  - **Quartz Composer.app/Contents/Frameworks**
- Put the **TOAQuartzComposerPlugIns.plugin** into the folder:
  - **/Library/Graphics/Quartz Composer Plug-Ins**

Disabling the TOA System Checker

> Close **all** ToolsOnAir applications on your Mac before disabling the TOA System Checker.

To develop, debug and deploy custom interface controlled graphics, it is required to set up a separate just:live or just:play system, which must **not** be used for an actual playout.

- Double click the **Disable TOA System Checker** app. Read and accept the dialog. A success dialog should appear shortly after.

Folder Structure & File Naming

To make a custom control interface work, this **folder structure** and **file naming** is required:

- **The graphic file:** the composition:builder or QTZ file can have any name.
- **The ".toaInterface" folder:** next to the graphic file it is mandatory to have a folder with the **same name as the graphic file**, followed by the suffix "**.toaInterface"**.
- **Content of the ".toaInterface" folder:** this folder must contain at least the file **toa.js** (which is initially empty, but required for runtime) and the file **index.html**.

How it works?

A custom control interface consists of a **HTML file** which incorporates **JavaScript** to transmit data which is entered via **HTML forms** (or other means). There is **one** JavaScript function which provides the functionality to **send** data to a graphic which is currently playing out. Please note that you **must** publish your desired Input Ports on the **Root Macro Patch** layer of your Quartz Composition.

```
toa.setInputPort('nameOfThePublishedInputPort',
valueWhichShouldBeTransmitted);
```

Once the graphic file is dropped into the Workbench of just:live/just:play and then played out, the custom control interface appears as a **floating window**. It is now ready to use.

What else?

Reopening a Closed Control Interface

Once the floating window of the custom control interface has been closed, it cannot be reopened anymore by default. A dummy Patch is required to show a button in the Inspector of just:live/just:play which can open up the window again.

- **For Quartz Composer compositions:** on the Root Macro Patch layer, create and publish an **Input Splitter** Patch (String type). Call it **CI_DUMMY**.
- **For composition:builder compositions:** there is a component in the Extras menu of the toolbar. Just add it to the Workbench and move it into a corner. The size does not matter and it is also invisible.

Using the same names for HTML Form IDs and QTZ Input Ports

An alternative approach to transmit data to the QTZ is to use JavaScript events (onChange, onBlur...) and the this object. To do so, use the same names for the QTZ input ports and the IDs of the HTML forms. Example:

```
<input type="text" id="textfieldMyTextField"
onChange="toa.setInputPort(this.id, this.value)">
```

In the **Root Macro Patch** layer of the QTZ there has to be an input port called **textfieldMyTextField**.

Accessing the Custom Interface with a Web Browser

You can access a custom interface with a web browser on any device (Mac, PC, tablet, smartphone), as long as your network is configured properly. To do so, enter the IP address of the host machine (i.e. the machine which plays out the graphic) and append ":5000" to the address in the web browser of your choice.

> To access the control interface on the local development machine, use **localhost:5000**.