INScore Web

Version 1.28

D. Fober

Centre national de création musicale

Contents

1 Introduction

Since version 1.27, the INScore engine is available as Javascript libraries:

- a WebAssembly [WASM] library providing all the services of the abstract INScore model,
- a Javascript library providing an HTML view of the INScore model.

The web environment provides a very different runtime context than a native application: it is much more modular; due to the absence of a 'concrete machine' a number of INScore primitives do not make sense in a web environment; finally it provides new rendering capabilities with CSS.

This document is intended to present the differences between the native and web versions of INScore. A special section is also devoted to the implementation of INScore Web in standalone HTML pages.

2 Behavioural differences

The OSC protocol is not supported in the Web version. As a result, the mode of communication with the INScore engine is different (see section 6) and may also depend on the application that uses this engine.

By default:

- the OSC output and error ports are redirected to the Javascript console.
- drag & drop works like in the native version: you can drop files or text to an INScore scene (an HTML div in the Web version).

The `log` window (address `/ITL/log`) is dependent on the host application. By default, input messages addressed to the `log` node are directed to the Javascript console.

2.1 Optional components

As the architecture of the web version is completely modular, the available objects depend on the host application: e.g. a page which wants to use objects in symbolic notation (type `gmn`) will have to include the Guido library. This architecture allows applications to be optimized to fit their needs. It also facilitates the extension of the INScore engine. The table 1 presents the current supported components.

Component	Name	Dependent types
Guido Engine	libGUIDOEngine.js	`gmn`, `gmnf`, `gmnstream`, `pianoroll`, `pianorollf`, `pianorollstream`
MusicXML library	libmusicxml.js	`musicxml`, `musicxmlf` use of MusicXML implies to have also the Guido Engine
Faust compiler	libfaust-wasm.js FaustLibrary.js	`faust`, `faustf`

Table 1: Components required by specific objects

2.2 Using files in a script

You can use file based objects in an INScore script but the file path is interpreted differently:

- when using an absolute path, it refers to the document root of the HTTP server,
- when using a relative path, it refers to the location of the HTML page

NOTE

Browsers infer a MIME type from the file extension and generally, download any file which extension is not recognized (this behavior depends on the browser you are using). It is therefore recommended to use a `.txt` extension for any textual resource with non-standard extension. For example, a `score.gmn` file could be renamed and used as `score.gmn.txt`.

3 Unsupported

3.1 Unsupported objects

The table 2 presents the objects that are not supported:

Type	Comment
`fileWatcher`	unsupported
`httpd`	unsupported server
`websocket`	unsupported server
Faust plugins	deprecated and redesigned (see section 5.2)
Gesture follower	unsupported

Table 2: Unsupported objects

The following components are not yet implemented:

- graphic signals objects: `graph`, `fastgraph`, `radialgraph`
- Pianoroll stream: `pianorollstream`
- Misc.: `grid`
- Memory image: `memimg`
- Sensors: `acceleromter`, `gyroscope`, `compass`, etc.

3.2 Unsupported messages

A number of messages are not supported in the web version, either because they do not make sense in the runtime context or because they cannot be implemented.

3.2.1 Common messages

- `export`, `exportAll`, `save`: not implemented
- `shear`, `dshear`: not implemented
- `effect`: `colorize`: not implemented
- `edit`: not implemented

3.2.2 Application messages

- `quit`: do not make sense in the web environment
- `rootPath`: not implemented
- `mouse`: not yet implemented
- `read`: not implemented
- `load`: not implemented
- `port`, `outport`, `errport`: do not make sense without OSC
- `get`: `guido-version` `musicxml-version`: not implemented.
  Note that the corresponding components are loaded, they print their version number to the Javascript console.

3.2.3 Application log window

As mentioned above, the `log` window (address `/ITL/log`) is dependent on the host application. By default, input messages addressed to the `log` node are directed to the Javascript console.

- `clear`: dependent on the host application (do nothing by default)
- `save`: not supported due to the lack of file system

- `foreground`: dependent on the host application (do nothing by default)
- `wrap`: dependent on the host application (do nothing by default)
- `scale`: dependent on the host application (do nothing by default)
- `write`: dependent on the host application (write to the Javascript console by default)

3.2.4 Scene messages

- `save`: not supported due to the lack of file system
- `foreground`: dependent on the host application (do nothing by default)
- `rootPath`: not yet implemented
- `load`: not yet supported
- `frameless windowOpacity`: not supported

3.2.5 Type specific messages

- `brushstyle`: not yet implemented
- Piano roll messages: not yet implemented
- Video `get` messages: not yet implemented
- SVG `animate`: not yet implemented
- Arc `close`: not yet implemented
- Line `arrows`: not yet implemented
- `debug` node: not yet implemented

Symbolic score:

- `columns rows`: not supported
- `pageFormat`: not yet implemented
- `get: pageCount systemCount`: not yet implemented

3.2.6 Synchronization

The following synchronisation modes are not yet supported:

- Stretch modes: `h`, `hv`

3.2.7 Events

The following events are not yet supported:

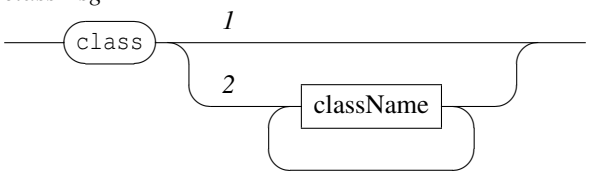
- Touch events: `touchBegin`, `touchEnd`, `touchUpdate`
- Url events: `success`, `error`, `cancel`
- Score event: `pageCount`
- `export`: not supported since the `export` message is not supported
- `endPaint`: not supported

4 Specific new messages

4.1 Leveraging CSS

Cascading Style Sheets [CSS] are a powerful way to control the appearance of elements on a web page. The Web version of INScore provides a specific `class` message to use CSS in parallel to the standard mechanisms. This message is supported by all the INScore objects.

classMsg



- 1: without argument, remove all class settings
- 2: set the CSS classes of an object

NOTE

If a `class` message has not the expected effect, it's likely because the CSS properties of the target object are set by the standard INScore mechanisms (like color, border, font-size, etc.). As the own attributes of an object have precedence over the class of the object, the properties of the class are then ignored. You can force the properties of the class by adding the CSS rule `!important` which will override all previous styling rules for that specific property.

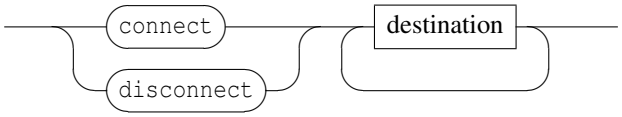
5 Audio objects

5.1 Audio objects

Audio objects are *connectable* objects i.e. objects which output can be connected to the input of another audio object. In the INScore model, the following objects are audio objects: `audio`, `video`, `faust` (see section 5.2) and `audioio` (see section 5.3).

Audio objects support the following messages:

audioMsgs

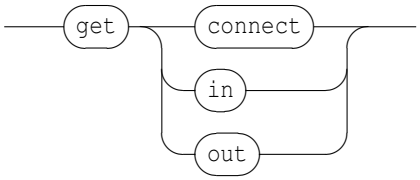


- `connect`: connect the outputs of the object to the destination inputs. The destination must be a another audio object. Inputs and outputs count must be the same on source and destination. The destination supports regular expressions.
- `disconnect`: disconnect the outputs of the object from the destination inputs. Note that errors (e.g. no existing connection with the destination) are silently ignored.

NOTE

The `connect` message assumes that the source and destination are located in the same hierarchy (i.e. they have the same parent).

audiogetMsgs



- `in`: gives the number of inputs of the audio object
- `out`: gives the number of outputs of the audio object

5.2 Faust objects

Faust is a functional programming language for sound synthesis and audio processing. Faust objects are available, providing that the Faust library has been loaded.

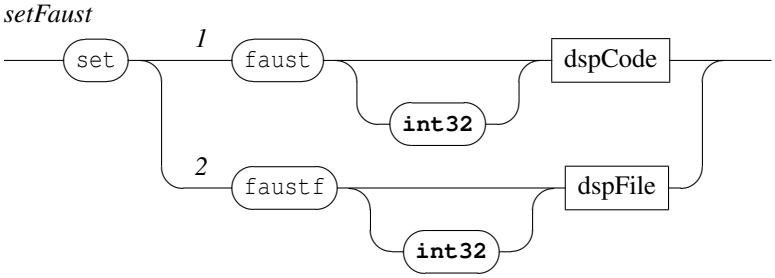
5.2.1 Creating a Faust object

The `faust` or `faustf` types must be used to create a Faust object.

NOTE

The `faust` type exists with the native version but to load a pre-compiled DSP. `faustdsp` and `faustdspf` types are not supported.

setFaust



The expected arguments of the `set` message are:

- an optional integer that indicates a number of voices used to create a polyphonic DSP (see section 5.2.4). Note that when present, a polyphonic DSP is created even if equal to 1.
- 1: Faust DSP code (see the Faust language for more information).
- 2: a Faust DSP file.

By default, a Faust DSP appears as a browsable block diagram.

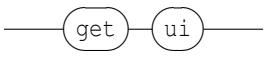
NOTE

The Faust language uses characters that have a special meaning in HTML: for example, with the split operator `<:`, the '<' character will be interpreted as an opening HTML tag and you should use HTML escapes (e.g. `<` instead of <). This is necessary for inline DSP code only, DSP files are not concerned.

5.2.2 Faust messages

Faust objects supports the audio objects messages plus an additional query message:

faustgetMsgs



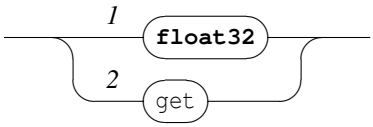
- `ui`: gives the Faust processor parameters (see section 5.2.3) i.e. for each parameter: its OSC address followed by the parameter UI type, a label, the minimum and maximum values and the parameter step.

5.2.3 Faust objects parameters

A Faust DSP code can declare UI elements that are used by *architecture files* to build controllers providing users with dynamic control of the DSP parameters. In INScore, DSP UI elements are used to extend the Faust object address space. For example, when a DSP code declares a UI element named 'Volume', a Faust object which address is `/ITL/scene/dsp` is extended as `/ITL/scene/dsp/Volume`.

Faust parameters support two types of messages:

faustParamMsgs



- 1: set the parameter value
- 2: gives the parameter value

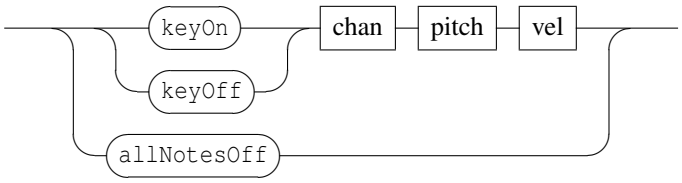
NOTE

When a parameter which type is `button` is set to 1, it is automatically reset to zero after a short time (corresponding at least to one audio buffer size).

5.2.4 Polyphonic objects

Polyphonic objects (i.e. Faust objects created using the optional voice number) support additional messages:

faustPolyMsgs

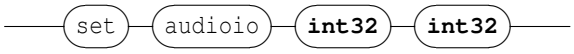


- `keyOn` and `keyOff` messages take 3 integer parameters: a MIDI channel, the note pitch and the velocity.
- `allNotesOff`: similar to MIDI all notes off message

5.3 Audio Input / Output

Audio input / output objects are provided as audio object (*see* sction 5.1) to the represent the physical audio inputs and outputs, in order to homogenize the connection process. An audio input / output type is `audioio`. It is created with a number of inputs and outputs as arguments.

audioioMsgs



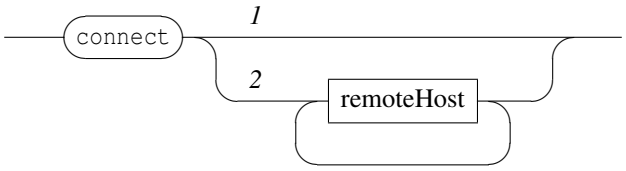
NOTE

The current Javascript implementation automatically creates an audio input objet named `audioInput` (if any) and an audio output object named `audioOutput` (if any). Thus the names `audioInput` and `audioOutput` are reserved and you should avoid to use them (unless audio connections are not needed).

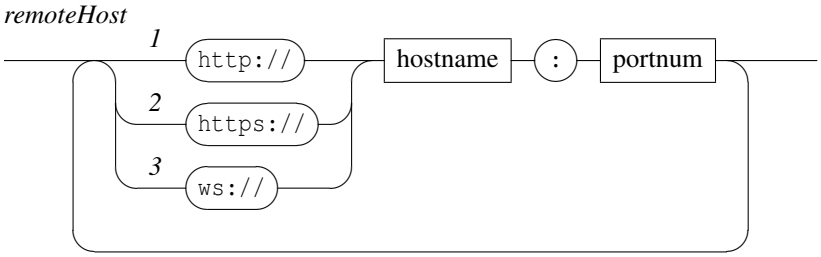
6 Communication scheme

INScore forwarding mechanism supports `http` and `ws` protocols since version 1.27. Since these protocols are connection based, a counterpart `connect` message is provided with the Web version.

connectMsg



- 1) removes the existing set of connections,
- 2) connect to a list of remote hosts. Once the connection is established, the local INScore engine may receive messages from the remote hosts. To establish the connection, the remote hosts must have set a forwarding mechanism using the same protocol with the same port number.



- 1) uses `http` as communication protocol,
- 2) uses `https`,
- 3) uses `websockets`.

6.1 Web messages format

Messages emitted by the `http` and `ws` forwarding mechanism and received by connected clients are encoded in JSON and transmitted as base64 encoded packets. The JSON format is the following:

```
{
   'id' :  number ,
   'method' :  'post' ,
   'data' :  textual inscore messages
}
```

- `id`: is a packet unique identifier (currently unused)
- `method`: value must be 'post'
- `data`: must contain a valid inscore script

Although the native version of INScore supports this format for the `http`, `https` and `ws` protocols, nothing prohibits another application to control INScore web pages, provided that the format described above is respected.