# The SysSon Platform

**Technical Report TR-2017-01-1**
**Institute of Electronic Music and Acoustics, Graz**
**(Status: in progress)**

Hanns Holger Rutz

January 2017

# 1 Adding an Offline Preprocessing Stage

The current workflow has been to experiment with matrix preprocessing directly in the IntelliJ IDE and outside of Mellite/SysSon. This is fine, because the IDE is much more powerful, and we have easier access to some of the utilities. However, the problem arises that we will need to make the results of these experiments available inside SysSon if we want to eventually allow the users to apply particular, newly developed preprocessing steps. We have done so with a separate menu-item to calculate anomalies, for example. We would now need to add another item for analysing the blobs. Obviously, this is not a scalable approach.

In the summer of 2016, I began to implement the next-generation FScape software, a toolkit for musical signal processing, using a UGen graph approach similar to ScalaCollider, but running offline. The back-end architecture uses the Akka-Stream framework, while the front-end API is very similar to ScalaCollider. This new FScape 2.x is already stable enough to use, as was demonstrated in various audio- and video-installations last year. Furthermore, it has a basic integration with SoundProcesses now. It is therefore an obvious choice to extend FScape with SysSon-specific UGens. A possible drawback is that the streams are weakly typed one dimensional signals (although channels can be bundled). There are a number of two-dimensional matrix and image processing UGens, but it requires that row size or width information must be explicitly passed into the respective UGens. Nevertheless, we estimate that it will be possible to implement the required UGens for SysSon matrices this way.

## 1.1 Implementation Steps

We have collected the required steps for a full implementation for processing matrices offline in SysSon below. As of January 26, the first five steps have been completed.

1. enhance SoundProcesses by "lazily" calculated objects
2. implement such a lazy object interface for FScape output
3. implement a caching mechanism for these objects
4. implement a simple SysSon matrix reader in FScape
5. make `Sonification` instance matrices available in FScape
6. add commonly required meta data UGens, such as rank, size, dimensional information
7. add dimensional operators, such as transposition
8. add the possibility to *output* matrices from FScape
9. make it possible to use these cached matrices as input to the real-time sonification
10. add UGens necessary to complete existing processes (anomalies, blobs)

### 1.1.1 Lazily Calculated Objects

TODO:

### 1.1.2 Lazily Objects in FScape

TODO:

### 1.1.3 Caching Objects

TODO:

### 1.1.4 Simple SysSon Matrix Reader UGen

TODO:

### 1.1.5 Accessing Sonification Instances

TODO:

### 1.1.6 Meta Data UGens

TODO:

### 1.1.7 Dimensional Operators

TODO:

### 1.1.8 Matrix Output

TODO:

### 1.1.9 Matrix Linkage from Offline to Real-Time

TODO:

### 1.1.10 Implementing Anomalies and Blobs

TODO: