

# The sadam Library

Siska Ádám

August 25, 2011

The sadam Library is a collection of externals written for Max 5<sup>1</sup>. This file contains an overview of these externals as well as a guide to install the objects and their documentation. You'll find the `sadam.stream` SDK and legal information at the end of this readme. Should you have any question not covered by this document, please contact me at `sales@sadam.hu`.

## 1 Overview

Name	Kind	Description
[sadam.base64]	Max	Base64 encoder & unencoder object <sup>2</sup> .
[sadam.empty]	Max	Detect/output empty symbols.
[sadam.envelopeGenerator]	Max	Expr-based envelope generator for [function] objects.
[sadam.float]	Max	Detect/output special floating point values.
[sadam.gcd]	Max	Compute the Greatest Common Divisor of two integers.
[sadam.lcm]	Max	Compute the Least Common Multiple of two integers.
[sadam.limits]	Max	Get minimum and maximum finite values of Max data types.
[sadam.lzo]	Max	Loseless data compression and decompression using the LZO library <sup>2,3</sup> .
[sadam.prime]	Max	Compute the closest prime numbers and the prime factorization of a positive integer.
[sadam.rapidXML]	Max	A RapidXML <sup>4</sup> wrapper for Max <sup>5</sup> .
[sadam.sortLists]	Max	Sort a set of lists simultaneously.
[sadam.split]	Max	Split incoming number sequence.
[sadam.stat]	Max	Get mean and standard deviation of a number sequence.

---

<sup>1</sup>Copyright © 1990–2008 Cycling '74/IRCAM. See <http://www.cycling74.com>

<sup>2</sup>The externals `sadam.base64` and `sadam.lzo` were commissioned by Prof. Georg Hajdu and the Co–Me–Di–A Project to serve the network music environment Quintet.net. For more information, see <http://www.quintet.net>.

<sup>3</sup>Using the LZO Library version 2.03 (30<sup>th</sup> April, 2008), Copyright © 1996–2008 Markus Franz Xaver Johannes Oberhumer. See <http://www.oberhumer.com/opensource/lzo>

<sup>4</sup>Using RapidXML version 1.13, Copyright © 2006–2009 Marcin Kalicinski. See <http://rapidxml.sourceforge.net/>

<code>[sadam.stream]</code>	Max	Read and write binary streams.
<code>[sadam.tcpReceiver]</code>	Max	Receive data from the network using the TCP protocol.
<code>[sadam.tcpSender]</code>	Max	Send data through the network using the TCP protocol.
<code>[sadam.udpReceiver]</code>	Max	Receive data from the network using the UDP protocol.
<code>[sadam.udpSender]</code>	Max	Send data through the network using the UDP protocol.
<code>[sadam.addSynth~]</code>	MSP	Additive synthesizer with time-dependent parameters <sup>6</sup> .
<code>[sadam.fofSynth~]</code>	MSP	Formant Synthesizer with time-dependent parameters <sup>6</sup> .
<code>[sadam.getSoundDescriptors~]</code>	MSP	Extract sound descriptors from an arbitrary sound <sup>6</sup> .
<code>[sadam.normalize~]</code>	MSP	An object that sets the gain of a source to the level of a reference signal.
<code>[sadam.peakExtractor~]</code>	MSP	Extract spectral peaks from a signal in real-time <sup>6</sup> .
<code>[sadam.phasor~]</code>	MSP	Phasor object with maximal resting state.
<code>[sadam.rand~]</code>	MSP	Band-limited random signal.
<code>[sadam.sgn~]</code>	MSP	Sign of a signal.
<code>[sadam.simpleAddSynth~]</code>	MSP	Additive synthesizer with constant parameters <sup>6</sup> .
<code>[sadam.standardMap~]</code>	MSP	A chaotic oscillator based on Chirikov's Standard Map <sup>7</sup> .
<code>[sadam.subSynth~]</code>	MSP	Subtractive synthesizer with time-dependent parameters <sup>6</sup> .
<code>[sadam.dom]</code>	MXJ	A Document Object Model (DOM) interface for Max <sup>5</sup> .
<code>[sadam.sax]</code>	MXJ	A Simple API for XML (SAX) interface for Max <sup>5</sup> .

## 2 Install

If you have an older version of the `sadam` Library installed, it is advised to uninstall the old version before installing the new one.

---

<sup>5</sup>The externals `sadam.rapidXML`, `sadam.dom` and `sadam.sax` were commissioned by Prof. Georg Hajdu to be included in MaxScore. For more information, see <http://www.algomusic.com/maxscore>.

<sup>6</sup>The externals `sadam.addSynth~`, `sadam.fofSynth~`, `sadam.getSoundDescriptors~`, `sadam.peakExtractor~`, `sadam.simpleAddSynth~` and `sadam.subSynth~` were commissioned by Prof. Johannes Kretz and the ZiMT and are part of the Klangpilot Project. They are not included in this release, instead, you can download them directly from the ZiMT Download Page at <http://www.mdw.ac.at/zimt/downloads-e.html>. These externals are currently undocumented but a detailed documentation will appear in a later version of the `sadam` Library.

<sup>7</sup>For details, see attached paper (from the 2<sup>nd</sup> Music in the Global Village Conference, Budapest, 2009).

## 2.1 Installing the Externals

The externals compiled for the Macintosh are located in the `mxo` folder while the ones for Windows are in the `mxw` folder. To install them, move the content of the respective folder (depending on your system) anywhere in your Max Search Path. It is a good practice however to keep all the (needed) files in a separate place of the `Cycling '74` folder<sup>8</sup> – for instance, by creating a `sadam` folder – to access them easily and to make later updates easier.

In addition, you need to move the file `sadamLib.jar` to the subfolder `java/lib/` of the `Cycling '74` folder in order to use the MXJ-based externals (currently `sadam.dom` and `sadam.sax`).

## 2.2 Installing the Documentation

The documentation consists of the Max Help Files and the References.

To install the Help Files, move all files from the `maxhelp` folder to anywhere inside the `Cycling '74` folder – it is a good practice though to put the Help Files to the same place as the externals themselves to make later updates easier.

As `Cycling '74` has not officially published the method to create Reference documentation for third-party externals, the installing method for the References might not work for later versions of Max (it works fine for Max 5.1.8, though). First, you need to locate the `refpages` folder of Max<sup>9</sup>. Then follow these steps:

1. Move the contents of `max-ref` to the `max-ref` subfolder of `refpages` and the contents of `msp-ref` to the `msp-ref` subfolder of `refpages`.
2. Move the contents of `max-images` to the `images` subfolder of the `max-ref` subfolder of `refpages` and the contents of `msp-images` to the `images` subfolder of the `msp-ref` subfolder of `refpages`.
3. Open the file `_c74_contents.xml` in the `max-ref` subfolder of `refpages` and insert the following lines (it is a good practice to put them before the first or after the last `refpage` entry):

```
<refpage name='sadam.base64.maxref.xml' />
<refpage name='sadam.empty.maxref.xml' />
<refpage name='sadam.envelopeGenerator.maxref.xml' />
<refpage name='sadam.float.maxref.xml' />
<refpage name='sadam.gcd.maxref.xml' />
<refpage name='sadam.lcm.maxref.xml' />
<refpage name='sadam.limits.maxref.xml' />
<refpage name='sadam.lzo.maxref.xml' />
<refpage name='sadam.prime.maxref.xml' />
<refpage name='sadam.rapidXML.maxref.xml' />
<refpage name='sadam.sortLists.maxref.xml' />
<refpage name='sadam.split.maxref.xml' />
<refpage name='sadam.stat.maxref.xml' />
<refpage name='sadam.stream.maxref.xml' />
```

---

<sup>8</sup>Macintosh: `/Applications/Max5/Cycling '74`

Windows: `C:\Program Files\Cycling '74\Max 5.0\Cycling '74`

<sup>9</sup>Currently `/Applications/Max5/patches/docs/refpages` on Macintosh,  
`C:\Program Files\Cycling '74\Max 5.0\patches\docs\refpages` on Windows.

```

<refpage name='sadam.tcpReceiver.maxref.xml' />
<refpage name='sadam.tcpSender.maxref.xml' />
<refpage name='sadam.udpReceiver.maxref.xml' />
<refpage name='sadam.udpSender.maxref.xml' />
<refpage name='sadam.dom.maxref.xml' />
<refpage name='sadam.sax.maxref.xml' />

```

4. Open the file `_c74_contents.xml` in the `msp-ref` subfolder of `refpages` and insert the following lines (it is a good practice to put them before the first or after the last `refpage` entry):

```

<refpage name='sadam.normalize~.maxref.xml' />
<refpage name='sadam.phasor~.maxref.xml' />
<refpage name='sadam.rand~.maxref.xml' />
<refpage name='sadam.sgn~.maxref.xml' />
<refpage name='sadam.standardMap~.maxref.xml' />

```

This procedure might need to be repeated each time after running the Max installer as the installer *might* overwrite the folders containing the documentation.

An easier way to install the reference documentation is by simply moving it to the `Cycling '74` folder. In this case, however, the *see also* links might not work properly.

## 2.3 Additional Resources

The `resources` folder contains the source code of the `sadam.lzo` object, including the source code of the LZO Library 2.03 itself. This is because the LZO Library is licensed under GPLv2 and therefore all derivative works (like the `sadam.lzo` external) must be also released using this license. To build the `sadam.lzo` external, you'll need the freely available Max SDK<sup>10</sup>, which is not allowed to be distributed with this package. After setting up the environment, you'll have to locate a few files in the SDK and link them to the `sadam.lzo` project files. For details, see the Documentation of the Max SDK<sup>11</sup>. It also contains an additional copy of the header file `sadam.stream.h`, which is needed for those developers who would like to build custom externals that can communicate with the binary streams represented by `sadam.stream` (see details in Section 5). There is a paper as well (written by myself) which was presented at the 2<sup>nd</sup> Music in the Global Village Conference (Budapest, 2009) and which explains the backgrounds of the `sadam.standardMap~` object for those who wish to use it.

## 3 Uninstall

To uninstall the `sadam` Library, remove all files you moved to the Max folders during installation and remove any related entries from the `_c74_contents.xml` files in the respective folders.

<sup>10</sup>Currently available at the website of Cycling '74: <http://www.cycling74.com>.

<sup>11</sup>Also available at Cycling '74.

## 4 Changelog

The different versions are identified by their release dates instead of version or build numbers.

- 2011-08-25:** — **Fixed** DLL dependency bug on Windows.
- **Added** `sadam.empty`.
  - **Added** `sadam.limits`.
  - **Added** `sadam.rapidXML`.
  - **Added** `sadam.sortLists`.
  - **Added** `sadam.split`.
  - **Added** `sadam.dom`.
  - **Added** `sadam.sax`.
  - **Updated** `sadam.prime`: Prime factorization added.
  - **Updated** `sadam.standardMap~`: Frequency can be changed by user.
  - **Fixed** `sadam.float`: Boolean outlets now send boolean values.
- 2010-12-07:** — First official release.

## 5 Writing stream-aware externals

This short SDK lets you write third-party externals that could access or modify data contained by a `sadam.stream`. It assumes that you are already familiar with the C language and the Max SDK itself (so that you know how to build a third-party external in C for Max). The `sadam.stream` objects use the `globalsymbol` mechanism and the notification system presented by the Max SDK Documentation. `sadam.stream` stores the bytes as a C++ vector (part of the STL Library), this is the container type which will hold any data queried from the stream and this is the container which you must use if you would like to insert data in your stream. As you will see, it is possible to avoid the usage of a vector by sending and/or querying the contents of the stream byte-by-byte, but this is not an efficient, therefore not a recommended way to go.

By including the `sadam.stream.h` header file you will get some common strings used by a stream. If you will not compile your code with a C++ compiler, you will need to call the `sadam_stream_initcommonsymbols` function somewhere in your code (the best choice is in your main function) to set these common variables.

To catch any notifications of a stream, you'll have to write a method that responds to the `notify` message<sup>12</sup>:

```
void myobject_notify ( t_myobject * x, t_symbol * s,
                      t_symbol * msg, void * sender, void * data ) {

    if ( msg == stream_after_change ) {
        // Do some stuff with the changed stream
    } else if ( msg == stream_before_clear ) {
        // Do some stuff with the stream before clearing it
    }
    // Etc...
}
```

---

<sup>12</sup>See details in the Max SDK Documentation.

A stream will send four types of notifications, two of them can be disabled or enabled by the user (or by your code by invoking the proper method). Apart of this, Max will send notifications when a stream with a particular name was created or destroyed. These are the notifications you can get:

```
t_symbol * stream_binding
t_symbol * stream_unbinding
t_symbol * stream_before_change
t_symbol * stream_after_change
t_symbol * stream_before_clear
t_symbol * stream_after_clear
```

The first two will be sent by Max itself when a stream is bound or unbound to a symbol. This may or may not happen, depending on Max, at each creation/deletion of an instance of `sadam.stream`. The data field will contain a pointer to the `t_object` representing the stream that is being bound/unbound.

The next two will be sent when the stream is changed by an `add*`, `erase*`, `insert*` or `replace*` call to the stream and can be enabled or disabled, either by the user or by you, by setting the `notifyonchange` property of the stream. The data parameter will contain a pointer to a vector containing a copy of the stream before and after it has been changed.

The last two will be sent before and after the stream is being cleared. `stream_before_clear` will pass a pointer to a vector containing a copy of the stream in the data parameter, however, `stream_after_clear` will pass nothing.

To actually get the notifications, you'll need to register your object with the stream you'd like to listen to. This can be done by invoking the `globalsymbol_reference` method (defined in `ext_globalsymbol.h`):

```
void * stream = globalsymbol_reference ( x,
                                       "foo", stream_classname->s_name );
```

In the example above `x` is the pointer to your own object and `foo` is the name of the stream we wish to listen to. If the stream doesn't exist, we'd get a `NULL` pointer, otherwise we'd get an object pointer to the object holding the `foo` stream. Of course at some point we'll need to stop listening to the object (at least in our object freeing function). This is achieved by invoking another command:

```
globalsymbol_dereference ( x,
                           "foo", stream_classname->s_name );
```

Remember, for each call of `globalsymbol_reference`, a de-referencing must be called as well.

To read or modify the contents of a named `sadam.stream`, you'll have to invoke one of the internal methods of the class using `object_method` calls. The names of these methods are declared in `sadam.stream.h` and are quite self-explanatory. The parameters required by these calls are documented in the header file itself. For methods that get bytes from the stream (`getbyte` to get a single byte and `getarray` to get an array of bytes) you'll have to provide a pointer to an already initialized variable (either an unsigned char or a vector of unsigned chars). This will hold the return value of your query.

The pointer to the object containing your stream's data, which is required for an `object_method` call, is the one returned by `globalsymbol_reference`. If nonzero, you can invoke a call on one of its methods. Here are some examples:

```

unsigned char          myByte;
vector<unsigned char> myArray;
void * s = globalsymbol_reference ( x,
                                   "foo", stream_classname->s_name );

if ( ! s ) return;
myByte = 0xF2;
myArray.push_back ( 0x3E );
myArray.push_back ( 0x6D );
myArray.push_back ( 0x92 );
object_method ( s, stream_addbyte, myByte);
object_method ( s, stream_addarray, & myArray );
object_method ( s, stream_insertbyte, 2, myByte );
object_method ( s, stream_getbyte, 1, & myByte );
object_method ( s, stream_getarray, 0, 4, & myArray );
object_method ( s, stream_clear );
globalsymbol_dereference ( x,
                           "foo", stream_classname->s_name );

```

After running the above code, `myByte` will contain `0x3E` and `myArray` will be `0xF2, 0x3E, 0xF2, 0x6D, 0x92`, while the stream itself will be empty.

## 6 Copyright

As the external `sadam.lzo` is using the LZO Library, version 2.03 (April 30, 2008, Copyright © 1996–2008 Markus F. X. J. Oberhumer), which is licensed under GPLv2, this external is released under GPLv2. You will find a copy of this license in the folder containing the source code of the external as well as attached to the copy of the LZO library.

All other externals are licensed under the Creative Commons Attribution 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

The `sadam` Library comes free but without any kind of official support or warranty and the author has no responsibility for any damage, failure or any other kind of inconvenience that might result from the use of this Library. By using The `sadam` Library you automatically agree to the terms above.