



where software meets silicon



## Flexible & Scalable Media Switching

White Paper  
13<sup>th</sup> May 2011



Connecting People. Connecting Business.



# Contents

- 1 Media Switch Fabric ..... 1
- 2 Framework ..... 1
- 3 Components ..... 2
  - 3.1 Application ..... 2
  - 3.2 Paths ..... 2
  - 3.3 Path Nodes ..... 2
  - 3.4 Endpoints ..... 2
- 4 Transforms ..... 2
  - 4.1 Transform Interface ..... 2
  - 4.2 Shims ..... 2
- 5 Sample Buffers ..... 2
- 6 Path Execution ..... 3
  - 6.1 Timing ..... 3
  - 6.2 Load Monitoring ..... 3
  - 6.3 Multi-Core Load Balancing ..... 3
- 7 Memory management ..... 3
- 8 Source Code ..... 3
- 9 Statistics ..... 3
- 10 Licensing ..... 4
- 11 Roadmap ..... 4
- 12 Summary ..... 4

## 1 Media Switch Fabric

The VMX engineered by Emutex Ireland is a business grade IP PBX software phone system designed for use in small to medium sized enterprises. It will facilitate up to 128 phone extensions and provide enterprises with a complete suite of modern communication features.

At the heart of the VMX phone system is the Media Switch Fabric (MSF). The MSF is an extensible framework for connecting a variety of communication endpoints with differing time domains, codecs and frequencies together in any desired configuration. It is a scalable and modular system that allows extensibility through addition of third party modules or by using the rich set of built-in modules to build new functionality.

The MSF is a mature and tested framework that is optimised to run on Linux platforms. It is in daily use in many commercial VMX phone system installations handling analog, ISDN and VoIP traffic. It facilitates G.711, G.729 and iLBC codecs and provides jitter buffering, DTMF detection and generation, echo cancelation, mixing, gain control, voice prompts, call recording and many other features.

The VMX IP PBX software system is available under license. The MSF is supplied in binary form - source code can be made available on request to Emutex.

## 2 Framework

The VMX MSF is a software framework with an API to create, modify, execute and destroy data paths between communication endpoints. Paths consist of a sequence of nodes called transforms that are linked together. Transforms operate on media samples, which are structures containing media data and metadata about the media. Samples can grow or shrink from the head or the tail as required.

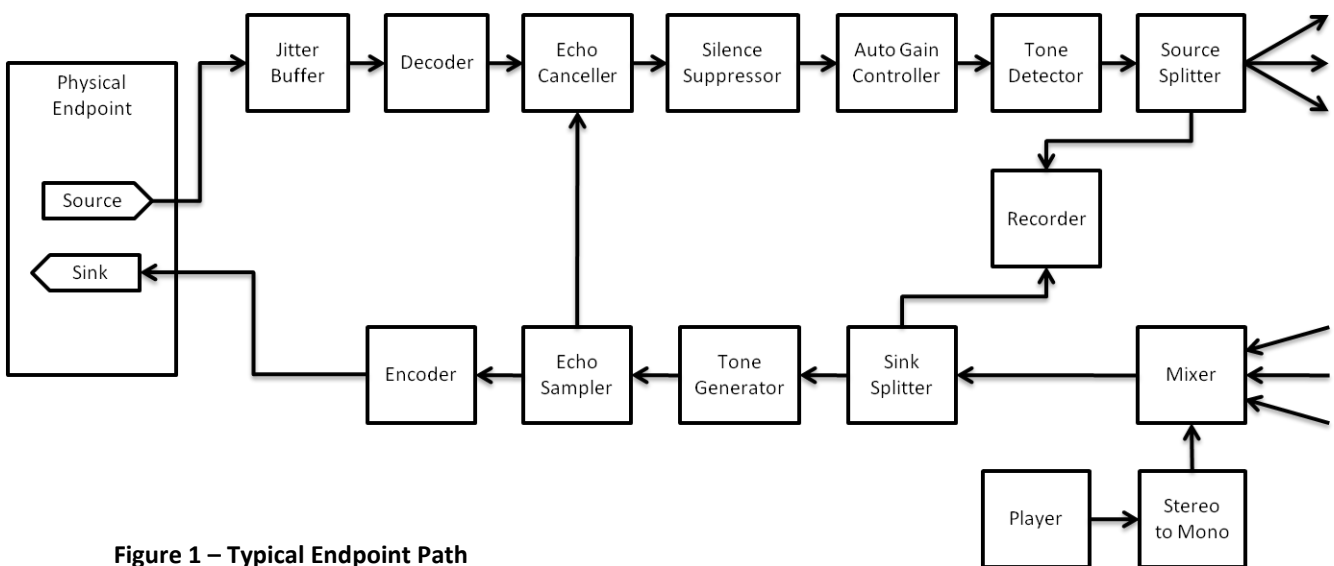


Figure 1 – Typical Endpoint Path

## 3 Components

### 3.1 Application

The MSF is a library with an API. In order to use it, it must be linked into an application that calls the API to create and then execute transforms on a path.

The application that executes the MSF typically does not perform any other activity, such as signalling, and runs at Linux real-time priority.

### 3.2 Paths

A path is a linked list of transform objects as shown in Figure 2. Each element in a path is executed in the order it appears in the list. The output of one transform is often the input to the next, but doesn't necessarily have to be.

Typically each endpoint has two paths - a sink and a source, although some endpoints only have a source (e.g. media player) and some only a sink (e.g. call recorder).

### 3.3 Path Nodes

Each node along a path is known as a transform. A transform performs a particular operation on a set of sample data. Depending on the transform, the input and the output data may be of the same format or may be different. Internally, the standard audio sample format is typically 8KHz or 16KHz 16-bit signed linear audio.

### 3.4 Endpoints

Many different types of endpoints are supported by the VMX MSF. An endpoint presents itself to the VMX as a pair of transforms, a sink and a source. The MSF has built-in endpoints for a variety of devices including PCI analog FXS and FXO cards, PCI BRI-ISDN and PRI-ISDN cards and RTP. Adding new endpoints is simply a matter of writing a wrapper to conform to the transform API and inserting into a path. The standardisation of the endpoint interface allows seamless connection of different endpoint types in one network. For example, there is no additional complexity required to connect a SIP phone using G.722 wideband codec to an analog TDM handset with a G.711u encoding.

## 4 Transforms

Transforms are nodes that are linked together to form a path. Each transform has inputs and outputs. There are five types of transform:

- A "sink" has one input and no outputs.

- A "source" has no inputs and one output.
- A "linear" transform has one input and one output.
- A "mixer" has multiple inputs and one output.
- A "splitter" has one input and multiple outputs.

The MSF validates API calls to build paths and uses the transform type to ensure that only valid connections are made.

Transforms can be built-in, statically linked external or dynamically linked external. Built-in transforms include G.711 encode/decode, mixer, splitter, AGC, VAD, tone-detect, tone-generator and FSK caller-id detection and generation,

Examples of statically linked transforms include SpanDSP echo canceller, G.722, G.726 and Howlertech G.729 (now owned by V-SYS UK).

Finally, the MSF API supports dynamic loading of transforms as shared objects. Transforms linked as shared objects conform to a standard API and register themselves with the MSF. Wrappers are provided in open source form for popular transforms that cannot be distributed with the MSF due to their GPL licensing. The OSLEC echo canceller is one example.

### 4.1 Transform Interface

All transforms must support a common interface. On being asked to register themselves, transforms must call the MSF register API with a structure containing function pointers for `init()`, `close()`, `execute()` and `control()` functions. These functions are defined in the `transform.h` header file. The control callback function functions similarly to an `ioctl` for a device driver and is used for out-of-band events. There is a minimum set of standard control functions that transforms are expected to support and optional control functions for statistics and control. Custom control messages are also supported. For example, the tone detector uses a control function to enable or disable tone clamping dynamically.

### 4.2 Shims

Shim transforms are used to convert from one time domain or frequency to another. For example, RTP which typically uses a 20ms packet time might use a shim to convert packets to and from 10ms for use by TDM interfaces.

## 5 Sample Buffers

Digitised media is passed between transforms in a sample buffer structure. This structure, as shown in Figure 2,

contains a byte array as well as pointers to data start, end, head and tail. It also contains metadata about the sample such as sample frequency, duration, encoding, etc.

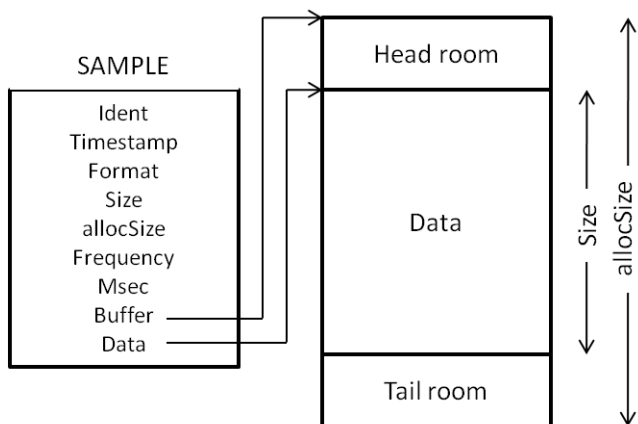


Figure 2 – Sample with Metadata

Samples are allocated and freed using API calls to the MSF. The MSF manages a fixed pool of samples. When samples are destroyed, they are returned to the pool. Samples are allocated with head room to allow addition of RTP and other protocol headers without having to be copied. Samples can also be cloned, chained and copied. Cloned samples are read-only. Chained samples are used to pass several samples together in one interval, e.g. a burst of RTP packets received by the de-jitter buffer.

## 6 Path Execution

### 6.1 Timing

All paths are traversed and executed at every time interval. Typically this is 10ms but can be configured to be any value.

The timing source is derived from hardware if possible. If a TDM device is present in the system, then a device driver using a blocking read() or ioctl() call is used to synchronise with the TDM hardware. If no TDM devices are installed, kernel timers are used to ensure the MSF is executed at a regular interval.

### 6.2 Load Monitoring

The time taken to run all paths and transforms is used to calculate the load average of the MSF. For example, if executing every transform on all paths takes 4ms and the base interval is 10ms then the system is considered to be 40% loaded. A warning is issued at a configurable limit and also the load value is provided through an API so higher level applications can make choices in response. For example, this feature is commonly used to limit the number

of concurrent calls using CPU intensive high compression codecs such as G.729.

## 6.3 Multi-Core Load Balancing

The MSF is capable of load balancing across multiple CPU cores if required. Although most of the MSF is single threaded for safety, there are a configurable number of path execution worker threads. Each of these threads has a synchronised start and stop per interval. After receiving a start event, each thread dequeues one path and executes it until there are no paths left to execute. In this way path execution is balanced across multiple cores. Since paths are largely independent of each other, locking is minimised and only required when handing over samples to another path or allocating or freeing samples.

## 7 Memory management

The MSF contains a custom memory management sub-system which allocates pre-determined memory block sizes to prevent fragmentation.

Safe versions of all memory and string functions are also provided to ensure buffer overruns cannot occur. For example, the MSF version of the sprintf() function automatically extends the memory block provided to prevent over-runs or truncations.

## 8 Source Code

The VMX MSF is developed in ANSI C and follows consistent coding standards to ensure quality. A built-in unit test framework allows standalone testing of key transforms.

The code currently runs on the Linux operating system but it is largely platform independent and can be ported to other operating systems or platforms if required.

## 9 Statistics

The MSF manages statistics on a per node and a per path basis. Each transform node supports a basic set of statistics including values such as samples processed, samples dropped, samples rejected, and execution count.

Timing statistics are also gathered and reported and allows quick determination of which transforms in a path are contributing most to the processing load.

## 10 Licensing

The Emutex MSF is a completely original work that is not derived from any other source code. It is available under a proprietary license from Emutex.

Since the framework is modular and extensible, it is possible to add modules with other licenses using a static or a dynamic link. Where a codec or other feature is available with an LGPL or other license that allows its use without affecting the rest of the framework, these modules may be linked against the MSF by Emutex or by licensees of the MSF.

Where a module or codec has a GPL or other license that would affect the MSF license, this module can be linked statically or dynamically against the MSF by a licensee only. Emutex will not distribute modules with GPL licenses. Licensees should note that inclusion of GPL licensed modules and codecs may have an impact on their rights to distribute products that use these them. For example, a wrapper for the OSLEC echo canceller is available from Emutex but Emutex does not distribute the OSLEC echo canceller. OSLEC has however been tested with the MSF and proven to work very well.

## 11 Roadmap

The MSF framework was designed from the outset to be flexible and extensible and will continue to evolve. Since no underlying assumptions have been made about the type of data that is being carried there is no limit to the possibilities for new use cases.

Wideband audio has recently been integrated into the MSF with the addition of the G.722 codec and associated shims to convert between 8KHz and 16KHz sampling. Other codecs and endpoints are continuously being evaluated for inclusion in the MSF.

Another candidate under research for future integration is on-demand video streaming. A media server built around the MSF could handle video on demand streaming, transcoding from differing sources, recording and playback using exactly the same structures as are used today for voice audio.

## 12 Summary

The MSF framework has successfully proven its ability to provide a robust and scalable media framework for voice

application. It is at the heart of the VMX IP-PBX and is already in everyday use at many commercial enterprises.

The combination of a powerful API for building interconnected paths in a scalable architecture, a standardised API for addition of transforms and endpoints and a proprietary license makes the VMX MSF a compelling licensing proposition for any company looking to build a media server or PBX. The portable code, simplicity of use of the API and small footprint make it especially compelling for embedded solutions.

EMUTEX Ltd.  
Axiom House,  
Raheen Business Park,  
Limerick, Ireland.

Tel: +353 (0)61 514 496  
Fax: +353 (0)61 513 059

General enquires: [info@emutex.com](mailto:info@emutex.com)  
Corporate web: [www.emutex.com](http://www.emutex.com)  
VMX web: [www.vmxphonesystems.com](http://www.vmxphonesystems.com)

