iMS SDK Version Update Record

March 2026: Code released on GitHub. Repositories include :

ims-lib: The core C++ library code, used directly or indirectly by everything else below.
imslib-python: The new Python library (not part of the SDK).
imslib-csharp, iMS_Studio, iMSIPConfig, ims_hw_server and ims_fw_upgrade.
(= software components that make up the SDK).

SDK v2-0-9

- Support for iCSA and iVCS throughout libraries and utilities, including iMS Studio
- Support for pulsed / duty cyled operation of RF output(s) iMS4 and iCSAs, including iMS Studio
- Merge in all of the codebase robustness improvements implemented during the Python development work
- The ims_hw_server application now ships with software hooks for C++, C#, Python, Ruby, PHP and Objective-C!
- All dependent libraries are brought up-to-date, improving security
- **A small number of code changes must be made to C++ applications when migrating from v1.x to v2.x iMS SDKs. These will be documented.**
- Library documentation is now online (https://isomet-corporation.github.io/ims-lib/)

SDK v1-8-12

- Frequency Resolution Parameter exposd in iMS Studio

Added support for new ImageFormat class (variable size Image data).Feature added to iMS HW Server. Allows-

- The number of RF Channels that are programmed by the Image
- The number of bytes used by: Frequency (range 1 - 4, default 2), Amplitude (range 1 - 2, default 1), * Phase (range 1 - 3, default 2), Sync Data (range 1 - 2, default 2).
- Whether amplitude and phase are sent as part of the image (if not, they can still be programmed using the Compensation LUT)
- Whether Sync Digital data is included with the Image
- Whether Sync Analog data is included with the image, and whether 1 or 2 channels are sent
- Combining multiple channels in pairs using 1 Image channel
- Combining all RF channels using 1 image channel

SDK v1-8-11

- Support for Clock Generator modes. When the iMS- is configured for internal image clock, this same clock is output on connector J11 for system synchronization
- Working Enhanced Tone Mode (ETM) for rev D/E Synths, plus an additional features called Frequency Fast Mod and Phase Fast Mod that use the two-level modulation feature of the AD9958 DDS chips to rapidly switch between two pre-programmed frequencies/phases using the profile pins.
- Both enhancements have been added into the hw server and Studio GUI.

SDK v1-8-10

- Fixes bug prevented the serial port scan disable to function correctly.

  Added a method for configuring the default connection scan using a configuration file on the filesystem; *C:\Users\<username>\AppData\Local\Isomet\iMS_SDK* > ***connection.xml***

  (Known issue using ***Enhanced Tone*** mode. See v1-6-0. To be corrected in future release)

SDK v1-8-9

- Fixes USB Handles resource leak

SDK v1-8-8

- Includes support for RF delay applied to both the first channel pair (Ch1/Ch2) and the second pair (Ch3/Ch4). Applies to rev-D only, firmware after v4.1.129
- iMS Studio also now detects a iMS2-HF board and disables unused amplitude control sliders displayed in the GUI.
- Corrects issue with linked amplitude control sliders in GUI (Sync Phase Pairs).

SDK v1-8-7

- Adds support for iMS4 rev-D, along with SignalPath class methods for X/Y channel delay and Sync digital output additional features (inversion, individual pulse/level).

SDK v1-8-6

- Increased sequence count from 65K to 3M for controller hardware configured with High Sequence Count Option only.

SDK v1-8-5

- Fix log flush issue that prevented applications from closing down cleanly.
- Disable COM port scan on iMS Studio (ims_hw_server) to speed startup of GUI.
- Synchronise code base with Linux library.
- IP Config application fixed on Win 7.

SDK v1-8-4

- Sequences now download in bulk mode which means any sequence with a large number of entries can be downloaded to the iMS system up to 100x faster **
- Fixes an issue that could have resulted in missed notifications from the iMS system to user software (e.g. Sequence Start / Finish).
- Improvements to the iMS Studio app. Panels on the right hand side now start up collapsed allowing you more workspace to modify image data. Hover over the tab to bring it into view and click on the drawing pin to fix each tab in place.
- Sync Data has moved from the Signal Path tab into its own tab, while there are new buttons in Signal Path for manual and automatic phase clear. Improvements have been made to the application start-up which should prevent the occasional crash on opening iMS Studio.

SDK v1-6-0

Legacy version to enable use of ***Enhanced Tone*** mode
(Known issue to be corrected in future release)

** Existing C++ software will continue to download sequences in the former “slow” mode.
To use the new fast download facility, the code must include a *SequenceDownloadSupervisor* which monitors the download events (similar to the process used for Image download) and reports to the application when the sequence download has completed OK.

Pseudo code with a ***WaitForSequenceDownload()*** **function** is provided in Appendix-A to illustrate the point.

Requires Firmware update 02020073 (FW v2-2-73) or later

It is necessary to upgrade the Controller firmware.
This will provide general speed enhancements, in addition the sequence download improvements described above.

After installing the new SDK, drag the following file:

C:\Program Files\Isomet\iMS_SDK\v1.8.4\data\fw\44332\Q0910A-02020073.mcs

onto the upgrade script:
C:\Program Files\Isomet\iMS_SDK\v1.8.4\utils\ims_fw_upgrade\upgrade_me.bat

Power cycle the iMS4 when complete.

Users of the **iMS2-HF** should also upgrade using the file at:

C:\Program Files\Isomet\iMS_SDK\v1.8.4\data\fw\57686\Q0915A-01010023.mcs

Appendix A: WaitForSequenceDownload() function

```
#include "ImageOps.h"
#include "LibVersion.h"

using namespace iMS;
class SequenceDownloadSupervisor : public IEventHandler
	{
	private:
		std::atomic<bool> m_downloading{ true };
		std::atomic<bool> m_error{ false };
	public:
		void EventAction(void* sender, const int message, const int param)
		{
			switch (message)
			{
			case (DownloadEvents::DOWNLOAD_FINISHED): std::cout << "Sequence
Download Finished!" << std::endl; m_downloading.store(false); break;
			case (DownloadEvents::DOWNLOAD_ERROR): std::cout << "Download error
on message " << param << std::endl;  m_downloading.store(false);  m_error.store(true);  break;
			case (DownloadEvents::DOWNLOAD_FAIL_TRANSFER_ABORT): std::cout <<
"Download failed: " << param << std::endl;  m_downloading.store(false); m_error.store(true);  break;
			}
		}
		bool Busy() const { return m_downloading.load(); };
		bool Error() const {
			return m_error.load();
		}
		void Reset() {
			m_downloading.store(true);
			m_error.store(false);
		}
	};

bool WaitForSequenceDownload(IMSSystem& ims, const ImageSequence& seq, bool
fast_download)
{
	std::unique_ptr<SequenceDownload> dl = std::make_unique<SequenceDownload> (ims,
seq);
	SequenceDownloadSupervisor ds;

	dl->DownloadEventsubscribe(DownloadEvents::DOWNLOAD_FINISHED, &ds);
	dl->DownloadEventsubscribe(DownloadEvents::DOWNLOAD_FAIL_TRANSFER_ABORT, &ds);
	dl->DownloadEventsubscribe(DownloadEvents::DOWNLOAD_ERROR, &ds);

	// Confirm that the linked iMS SDK supports the fast download process.
	// If it does, the library itself will check that the iMS firmware also supports it.
	// If either check fails, only the slower download process can be used.
	if (!LibVersion::HasFeature("FAST_SEQUENCE_DOWNLOAD")) {
		fast_download = false;
	}
```

```
	if (!dl->Download(fast_download)) {
		std::cout << "Error starting download" << std::endl;
		dl->SequenceDownloadEventUnsubscribe(DownloadEvents::DOWNLOAD_FINISHED,
&ds);
		dl-
>SequenceDownloadEventUnsubscribe(DownloadEvents::DOWNLOAD_FAIL_TRANSFER_ABORT,
&ds);
		dl->SequenceDownloadEventUnsubscribe(DownloadEvents::DOWNLOAD_ERROR,
&ds);
		return false;
	}

	while (ds.Busy() && fast_download) {
		std::this_thread::sleep_for(std::chrono::milliseconds(50));
	}

	dl->SequenceDownloadEventUnsubscribe(DownloadEvents::DOWNLOAD_FINISHED, &ds);
	dl-
>SequenceDownloadEventUnsubscribe(DownloadEvents::DOWNLOAD_FAIL_TRANSFER_ABORT,
&ds);
	dl->SequenceDownloadEventUnsubscribe(DownloadEvents::DOWNLOAD_ERROR, &ds);

	return !(ds.Error());
}

int main(int argc, char* argv)
{

	// Connect to iMS
	// Create and Download Images
	// Create your SequenceEntries and Sequences

	SequenceSupervisor seqs;
	SequenceManager seqMgr(myiMS);

	seqMgr.SequenceEventSubscribe(SequenceEvents::SEQUENCE_ERROR, &seqs);
	seqMgr.SequenceEventSubscribe(SequenceEvents::SEQUENCE_FINISHED, &seqs);
	seqMgr.SequenceEventSubscribe(SequenceEvents::SEQUENCE_START, &seqs);

	seqMgr.QueueClear();

	for (int n=0; n<total_seqs; n++) {
		std::cout << "Downloading Sequence " << n << std::endl;
		auto start = std::chrono::steady_clock::now();
		if (!WaitForSequenceDownload(myiMS, seq[n], true)) {
			std::cout << "Aborting.. <press any key>" << std::endl;
			char c;
			std::cin >> c;
			return 0;
		}
```

```
                auto duration = std::chrono::duration_cast<std::chrono::milliseconds>
                        (std::chrono::steady_clock::now() - start);
                start = std::chrono::steady_clock::now();
                std::cout << "Sequence " << n << " Load took " << duration.count() << " msec" <<
std::endl;
        }

        std::cout << seqMgr.QueueCount() << " sequence(s) in queue" << std::endl;

        // Start Sequence Queue

        return 0;
}
```