DTMediaRead Programmer's Interface

www.drastic.tv
February 9, 2024

Copyrights and Trademark Notices

General

ARRI AG – ARRI, Arri T-Link, and Alexa are registered trademarks of the ARRI Group

ASSIMILATE® Inc. - Assimilate SCRATCH and Assimilate SCRATCH Lab are either trademarks or registered trademarks of ASSIMILATE® Inc. or its subsidiaries in the United States and/or other countries.

ATI TECHNOLOGIES ULC - ATI is a trademark of ATI TECHNOLOGIES ULC

Autodesk, Inc. - Autodesk, Discreet, Flame, Flare, Smoke, Lustre, Maya, and Moxion are either trademarks or registered trademarks of Autodesk, Inc. or its subsidiaries in the United States and/or other countries.

Avid: Avid Media Composer®, Avid MediaCentral®, Avid Interplay®, and Avid NewsCutter® are either trademarks or registered trademarks of Avid Technology, Inc. or its subsidiaries in the United States and/or other countries.

Blackmagic: DaVinci Resolve, DaVinci Fusion, UltraStudio, DeckLink, Intensity Pro 4K, UltraScope, and RED are either trademarks or registered trademarks of Blackmagic Design Pty. Ltd. or its subsidiaries in the United States and/or other countries.

Bluefish444: Bluefish444, IngeSTore, Symmetry, Kronos, Epoch, Epoch:Neutron, Fury, Lust, Vengeance HD, Deepblue, Envy SD, and Epoch:SuperNova are trademarks of Bluefish Technologies

Boris FX, Inc. - Boris FX, Sapphire, and Silhouette are trademarks of Boris FX, Inc.

CANON KABUSHIKI KAISHA - CANON is a trademark of CANON KABUSHIKI KAISHA

Changsha Kiloview Electronics Co., Ltd - KILOVIEW is a trademark of Changsha Kiloview Electronics Co., Ltd

CineSys LLC – CineSys is a registered trademark of CineSys LLC.

Cisco Systems, Inc. - Cisco, and Webex are registered trademarks of Cisco Systems, Inc.

Cloudfirst Technology Solutions Inc. - Cloudfirst is a registered trademark of Cloudfirst Technology Solutions Inc.

Codex Corporation - CODEX and Action Cam are trademarks of Codex Corporation

Comtrol Corporation - Comtrol is a registered trademark of Comtrol Corporation

ConnectX, Inc - CONNECTX is a trademark of ConnectX, Inc

CoreCodec, Inc. - MATROSKA is a trademark of CoreCodec, Inc.

Corel Corporation - Pinnacle is a registered trademark of Corel Corporation

CORSAIR MEMORY, INC. - ELGATO is a trademark of CORSAIR MEMORY, INC.

Digital Vision World - Digital Vision World is an operating brand of BlissTek Ltd. BlissTek Ltd. Digital Vision Nucoda is either a trademark or registered trademark of BlissTek Ltd. or its subsidiaries in England, Wales, and/or other countries.

DIGITNOW! - Digitnow is a trademark of DIGITNOW!

Docker Inc. - DOCKER is a trademark of Docker, Inc.

Dolby: Dolby, Dolby Vision, the double-D symbol, and Millicast are registered trademarks of Dolby Laboratories.

Drastic Technologies: 2110Scope, 4KScope, ccConvert, Drastic Technologies, DrasticPreview, FlowCaster, HDRScope, Media File Scanner, MediaNXS, MediaReactor, MediaReactor Workstation, MR Lite, ndiScope, Net-X-Code Channel, Net-X-Code Server, Net-X-Convert, Net-X-Proxy, Network Video Analyzer, NetXfer, NETXROUTER, QuickClip, sdiScope, SyncControl, TcCalc, videoQC Inspect, videoQC Pro, videoQC View, and videoQC Workstation are trademarks of Drastic Technologies Ltd. All other trademarks are the property of their respective owners.

DSC Labs - DSC Labs' CamBook, CamAlign, and ChromaDuMonde charts are trademarks or registered trademarks of DSC Labs

Dublin Core™ Metadata Initiative - "Dublin Core" is a protected under common law trademark of the Dublin Core™ Metadata Initiative.

Eastman Kodak Company - Cineon™ is a trademark of Eastman Kodak Company

Eaton Corporation plc - Eaton, Tripp Lite, and PowerAlert are registered trademarks of Eaton Corporation plc

Empress Media Asset Management (eMAM) – eMAM, and eMAMDirector are registered trademarks of Empress Media Asset Management (eMAM)

Epiphan - All Epiphan product names and logos are trademarks or registered trademarks of Epiphan

Evercast, LLC - EVERCAST is a trademark owned by Evercast, LLC

Evertz Technologies Limited - Evertz is a registered trademark of Evertz Technologies Limited

EVS Broadcast Equipment - EVS is a registered trademark of EVS Broadcast Equipment

Fabrice Bellard - FFmpeg is a trademark of Fabrice Bellard

Filestage GmbH - Filestage is a trademark of Filestage GmbH

FilmLight Ltd. - FilmLight and BaseLight are trademarks of FilmLight Ltd.

Fraunhofer IIS and Thomson Multimedia: MPEG Layer-3 audio coding technology licensed from Fraunhofer IIS and Thomson Multimedia.

Free Software Foundation (FSF) - Portions of this product are licensed under LGPL, governed by the GNU LESSER GENERAL PUBLIC LICENSE, published by the Free Software Foundation (FSF).

Ftrack AB - FTRACK is a trademark and brand of Ftrack AB

Gen Digital Inc. (formerly Symantec Corporation and NortonLifeLock) - Symantec, Symantec Endpoint Virtualization Suite, Sygate, Altiris, and Altiris Virtualization Agent are registered trademarks of Gen Digital Inc.

Google: YouTube, Google, Google Cloud, Google.meet.com, and Android are registered trademarks of Google LLC

GoPro, Inc. - Cineform® is a trademark or registered trademark of GoPro, Inc.

Grass Valley USA, LLC - Grass Valley®, GV®, the Grass Valley logo, and EDIUS® are trademarks or registered trademarks of Grass Valley USA, LLC, or its affiliated companies in the United States and other jurisdictions.

HaiVision Systems, Inc. - Haivision is a registered trademark of HaiVision Systems, Inc.

Harris Corporation - Harris, and Leitch Technology Corp. are registered trademarks of Harris Corporation

Hewlett Packard Enterprise Company - OpenGL is a registered trademark and the OpenGL SC logo is a trademark of Hewlett Packard Enterprise Company

Hewlett Packard Group LLC - HP is a trademark of HP Hewlett Packard Group LLC.

Ikegami Electronics (USA) Inc. - EditCam is a registered trademark of Ikegami Electronics (USA) Inc.

Indiecam GmbH - IndieCam is a registered trademark of Indiecam GmbH

INOGENI Inc - INOGENI® is a Registered Trademark and TOGGLE is a Trademark of INOGENI Inc

Institute of Electrical and Electronics Engineers - IRE is a trademark of the Institute of Electrical and Electronics Engineers

INTEL CORPORATION - INTEL is a trademark of INTEL CORPORATION

International Business Machines Corporation ("IBM") - IBM® is a trademark owned by International Business Machines Corporation ("IBM") and might also be trademarked or a registered trademark in other countries

Interactive Effects, Inc. - Piranha is a registered trademark of Interactive Effects, Inc.

I**O Industries Ltd.** - IO Industries is a trademark of IO Industries Ltd.

Iteris, Inc. - Odetics is a registered trademark of Iteris, Inc.

JVC KENWOOD CORPORATION - JVC is a trademark of JVC KENWOOD CORPORATION

Kinefinity Inc. - KINEFINITY is a trademark of Kinefinity Inc.

L3Harris Technologies, Inc. - Louth is a trademark of L3Harris Technologies, Inc.

Linus Torvalds - Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

Logitech International SA - LOGITECH is a trademark of Logitech International SA

Magic Lantern - Magic Lantern is a registered trademark of Magic Lantern

MAINCONCEPT GMBH - MAIN CONCEPT is a trademark of MAINCONCEPT GMBH

Marshall Electronics, Inc. - Marshall is a registered trademark of Marshall Electronics, Inc.

Matrox Electronic Systems, Ltd - Matrox and Matrox product names are registered trademarks and/or trademarks of Matrox Electronic Systems, Ltd.

MediaArea.net SARL - MediaInfo - Copyright© 2002-2013 MediaArea.net SARL. All rights reserved.

Meta Platforms, Inc - Facebook and Instagram are trademarks of Meta Platforms, Inc

Microsoft: Windows®, Video For Windows (VFW), DirectShow, Microsoft, Skype, Microsoft Azure, Microsoft Teams, Wave Mapper, Microsoft, Windows NT|2000|XP|XP Professional| Server 2003|Server 2008 |Server 2012, Windows 7, Windows 8, Media Player, Media Encoder, .Net, Internet Explorer, SQL Server 2005|2008|2012|2014, Windows Media Technologies and Internet Explorer are trademarks of Microsoft Corporation.

MPEG LA - MPEG LA licenses patent pools covering essential patents required for use of the MPEG-2, MPEG-4, IEEE

1394, VC-1, ATSC, MVC, MPEG-2 Systems, AVC/H.264 and HEVC standards.

Nanjing Magewell Electronics Co. - Magewell™, ULTRA STREAM® and (the MAGEWELL Logo) are trademarks or registered trademarks of Nanjing Magewell Electronics Co.

Netflix, Inc. - Netflix is a registered trademark of Netflix, Inc.

NewTek, Inc. - NDI, TriCaster, 3Play, TalkShow, Video Toaster, LightWave 3D, and Broadcast Minds are registered trademarks of NewTek, Inc.

Nokia Corporation - OSPREY is a trademark owned by Nokia Corporation

NVIDIA Corporation - NVIDIA, the NVIDIA logo, NVIDIA Quadro, Rivermax, BlueField2, PhysX, and NVIDIA RTX are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and/or other countries

Object Matrix Limited - ObjectMatrix, and Object Matrix are registered trademarks of Object Matrix Limited

Omneon Video Networks, Inc - Omneon is a trademark of Omneon Video Networks, Inc

ONVIF - the ONVIF primary trademark is the word, "ONVIF". This trademark has been registered in the United States, European Union, China, Japan and other countries throughout the world.

Oracle Corporation - Oracle®, Java, Front Porch Digital, and MySQL are registered trademarks of Oracle Corporation and/or its affiliates.

Panasonic Holdings Co., Ltd - Panasonic, and Varicam are trademarks of Panasonic Holdings Co., Ltd

Pioneer Corporation - Pioneer is a registered trademark of Pioneer Corporation

RE:Vision Effects, Inc. - RE:Vision Effects is a registered trademark of RE:Vision Effects, Inc.

Red Hat, Inc. - Red Hat, and the Red Hat logo are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries

QT: The Qt Toolkit is copyright by The Qt Company and/or its subsidiary(-ies) and other contributors. The Qt Toolkit is used under the terms of the GNU Lesser General Public License v. 3 and the GNU Lesser General Public License v. 2.1 (both jointly "LGPL"). On each supported platform, the Tool dynamically links to the unmodified Qt libraries, as provided by the Qt Project in the pre-compiled binary format. In compliance with LGPL, all the relevant information about downloading, installing, and building the

Qt Toolkit from sources is available from http://www.drastic.tv. As there have been no modifications, the main source of the information and most of the web links provided here come from the Qt Company's website.

Shenzhen Yunlang Technology Co., Ltd. - MOKOSE is a trademark of Shenzhen Yunlang Technology Co., Ltd.

Sigma Design Company, LLC - Sigma Design is a registered trademark of Sigma Design Company, LLC

Snell & Wilcox Limited - SNELL & WILCOX, and Quantel are trademarks owned by Snell & Wilcox Limited

Society of Motion Picture and Television Engineers - SMPTE is a trademark of Society of Motion Picture and Television Engineers.

SoftNI Corporation – SoftNI is a trademark of SoftNI Corporation

Sony Corporation – Sony, Sony DVD Architect, DVD, Catalyst, and Vegas are trademarks of Sony Corporation and/or its affiliates.

Streambox Inc. - Streambox is a trademark of Streambox Inc.

Technicolor Creative Studios SA - Technicolor is a trademark of Technicolor Creative Studios SA

TechSmith Corporation - CAMTASIA STUDIO is a trademark of TechSmith Corporation

Tektronix, Inc. - Tektronix® and all identified Tektronix trademarks and logos are the property of Tektronix, Inc. or its wholly-owned subsidiaries

Telestream, LLC - Telestream, is a registered trademark, and MacCaption and CaptionMaker are trademarks of Telestream, LLC

The Apache Software Foundation (ASF) - Apache is a registered trademark of The Apache Software Foundation

The Foundry Visionmongers Ltd. - Nuke™ is a trademark of The Foundry Visionmongers Ltd.

The Perl Foundation - Perl and the Perl logo are trademarks of The Perl Foundation

Trend Micro Inc. - TrendMicro, and TrendMicro System Protection and registered trademarks of Trend Micro Inc.

Truevision, Inc - TARGA is a registered trademark of Truevision, Inc

Twitch Interactive, Inc - TWITCH, the TWITCH Logo, the Glitch Logo, and/or TWITCHTV are trademarks of Twitch Interactive, Inc. or its affiliates.

VideoLAN Non-profit Organization - VideoLAN, VLC, VLC media player and x264 are trademarks internationally registered by the VideoLAN non-profit organization

Vision Research, Inc - PHANTOM is a trademark of Vision Research, Inc

Weisscam GmbH - Weisscam is a trademark and brand of Weisscam GmbH

Wizards of OBS, LLC – UNIX, OBS, Open Broadcast Software, the OBS logo, and OBS Studio are trademarks of Wizards of OBS, LLC (The Company)

Wowza Media Systems, LLC - Wowza is a trademark of Wowza Media Systems, LLC

Xceed Software Inc. - Xceed DataGrid for JavaScript, Xceed Ultimate ListBox for Silverlight, Xceed DataGrid for Silverlight, Xceed DataGrid for WPF, Xceed Grid for .NET, Xceed Zip for .NET, Xceed Real-Time Zip for Silverlight, Xceed Upload for Silverlight, Xceed Zip Compression Library, Xceed FTP for .NET, Xceed Chart for .NET, Xceed Chart for ASP.NET, Xceed SmartUI for .NET, Xceed SmartUI, Xceed Encryption Library, Xceed Binary Encoding Library, Xceed Streaming Compression Library, Xceed Streaming Compression for .NET, Xceed Zip for .NET Compact Framework, Xceed Ultimate Suite, Xceed Data Manipulation Suite, Xceed Absolute Packager are trademarks of Xceed Software Inc.

Zapex Technologies - Zapex is a registered trademark of Zapex Technologies

Zhang Haijun - RYBOZEN is a trademark of Zhang Haijun

Ziflow Limited - Ziflow is a trademark of Ziflow Limited

Zixi - Zixi Software and any logos or icons identifying Zixi and the Zixi Software are trademarks of Zixi.

ZLIB: The ZLIB Compressed Data Format Specification is Copyright (C) 1995-2013 Jean-Loup Gailly and Mark Adler.

Zoom Video Communications, Inc. - Zoom and the Zoom logo are trademarks of Zoom Video Communications, Inc.

x264 LLC: The product is manufactured by Drastic Technologies under license from x264 LLC.

LGPL: Portions of this product are licensed under LGPL, governed by the following license:

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

0. Additional Definitions.

As used herein, “this License” refers to version 3 of the GNU Lesser General Public License, and the “GNU GPL” refers to version 3 of the GNU General Public License.

“The Library” refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An “Application” is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A “Combined Work” is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the “Linked Version”.

The “Minimal Corresponding Source” for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The “Corresponding Application Code” for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

- a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or
- b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice,

provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

- a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

- a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the Combined Work with a copy of the GNU GPL and this license document.
- c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.
- d) Do one of the following:
  - 0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.
  - 1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.
- e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

- a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.
- b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.

MPEG Disclaimers

MPEGLA MPEG2 Patent

ANY USE OF THIS PRODUCT IN ANY MANNER OTHER THAN PERSONAL USE THAT COMPLIES WITH THE MPEG-2 STANDARD FOR ENCODING VIDEO INFORMATION FOR PACKAGED MEDIA IS EXPRESSLY PROHIBITED WITHOUT A LICENSE UNDER APPLICABLE PATENTS IN THE MPEG-2 PATENT PORTFOLIO, WHICH LICENSE IS AVAILABLE FROM MPEG LA, LLC, 4600 S. Ulster Street, Suite 400, Denver, Colorado 80237 U.S.A.

MPEGLA MPEG4 VISUAL

THIS PRODUCT IS LICENSED UNDER THE MPEG-4 VISUAL PATENT PORTFOLIO LICENSE FOR THE PERSONAL AND NON-COMMERCIAL USE OF A CONSUMER FOR (i) ENCODING VIDEO IN COMPLIANCE WITH THE MPEG-4 VISUAL STANDARD ("MPEG-4 VIDEO") AND/OR (ii) DECODING MPEG-4 VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL AND NON-COMMERCIAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION INCLUDING THAT RELATING TO PROMOTIONAL, INTERNAL AND COMMERCIAL USES AND LICENSING MAY BE OBTAINED FROM MPEG LA, LLC. SEE HTTP://WWW.MPEGLA.COM.

MPEGLA AVC

THIS PRODUCT IS LICENSED UNDER THE AVC PATENT PORTFOLIO LICENSE FOR THE PERSONAL USE OF A CONSUMER OR OTHER USES IN WHICH IT DOES NOT RECEIVE REMUNERATION TO (i) ENCODE VIDEO IN COMPLIANCE WITH THE AVC STANDARD ("AVC VIDEO") AND/OR (ii) DECODE AVC VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSED TO PROVIDE AVC VIDEO. NO LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION MAY BE OBTAINED FROM MPEG LA, L.L.C. SEE HTTP://WWW.MPEGLA.COM.

MPEG4 SYSTEMS

THIS PRODUCT IS LICENSED UNDER THE MPEG-4 SYSTEMS PATENT PORTFOLIO LICENSE FOR ENCODING IN COMPLIANCE WITH THE

MPEG-4 SYSTEMS STANDARD, EXCEPT THAT AN ADDITIONAL LICENSE AND PAYMENT OF ROYALTIES ARE NECESSARY FOR ENCODING IN CONNECTION WITH (i) DATA STORED OR REPLICATED IN PHYSICAL MEDIA WHICH IS PAID FOR ON A TITLE BY TITLE BASIS AND/OR (ii) DATA WHICH IS PAID FOR ON A TITLE BY TITLE BASIS AND IS TRANSMITTED TO AN END USER FOR PERMANENT STORAGE AND/OR USE. SUCH ADDITIONAL LICENSE MAY BE OBTAINED FROM MPEG LA, LLC. SEE HTTP://WWW.MPEGLA.COM FOR ADDITIONAL DETAILS.

Drastic Technologies Limited Warranty and Disclaimers

Drastic Technologies Ltd (the Company) warrants to the original registered end user that the product will perform as stated below for a period of ninety (90) days from the date of licensing or; in the case of hardware, for a period matching the warranty period offered by the original manufacturer of said equipment.

Hardware and Media—The Product hardware components, if any, including equipment supplied but not manufactured by the Company but NOT including any third party equipment that has been substituted by the Distributor or customer for such equipment (the "Hardware"), will be free from defects in materials and workmanship under normal operating conditions and use.

Warranty Remedies

Your sole remedies under this limited warranty are as follows: Hardware and Media—The Company will either repair or replace (at its option) any defective Hardware component or part, or Software Media, with new or like new Hardware components or Software Media. Components may not be necessarily the same, but will be of equivalent operation and quality.

Software Updates

Except as may be provided in a separate agreement between Drastic Technologies and You, if any, Drastic Technologies is under no obligation to maintain or support the Software and Drastic Technologies has no obligation to furnish you with any further assistance, technical support, documentation, software, update, upgrades, or information of any nature or kind.

Restrictions and Conditions of Limited Warranty

This Limited Warranty will be void and of no force and effect if (i) Product Hardware or Software Media, or any part thereof, is damaged due to abuse, misuse, alteration, neglect, or shipping, or as a result of service or modification by a party other than the Company, or (ii) Software is modified without the written consent of the Company.

Limitations of Warranties

THE EXPRESS WARRANTIES SET FORTH IN THIS AGREEMENT ARE IN LIEU OF ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. No oral or written information or advice given by the Company, its distributors, dealers or agents, shall increase the scope of this Limited Warranty or create any new warranties.

Geographical Limitation of Warranty—This limited warranty is valid only within the country in which the Product is purchased/licensed.

Limitations on Remedies—YOUR EXCLUSIVE REMEDIES, AND THE ENTIRE LIABILITY OF Drastic Technologies Ltd WITH RESPECT TO THE PRODUCT, SHALL BE AS STATED IN THIS LIMITED WARRANTY. Your sole and exclusive remedy for any and all breaches of any Limited Warranty by the Company shall be the recovery of reasonable damages which, in the aggregate, shall not exceed the total amount of the combined license fee and purchase price paid by you for the Product.

Damages

Drastic Technologies Ltd SHALL NOT BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF YOUR USE OR INABILITY TO USE THE PRODUCT, OR THE BREACH OF ANY EXPRESS OR IMPLIED WARRANTY, EVEN IF THE COMPANY HAS BEEN ADVISED OF THE POSSIBILITY OF THOSE DAMAGES, OR ANY REMEDY PROVIDED FAILS OF ITS ESSENTIAL PURPOSE.

Further information regarding this limited warranty may be obtained by writing:

Drastic Technologies Ltd
523 The Queensway, Suite 201
Toronto, ON, M8V 1J7
Telephone: (416) 255-5636

Introduction

The DTMediaRead interface is designed to give programmers a simple yet powerful means of access to all of the still, video, streaming and audio formats that Drastic's MediaReactor, videoQC, Net-X-Code Server, and Drastic DDR software is able to read. This document describes the various methods and properties exported by DTMediaRead.

There are two ways to access the DTMediaRead API:

- ActiveX via direct 32 bit, or 32/64 bit RPC *(deprecated)*
- Direct link to Windows 32, Windows 64, macOS 32/64 and Linux 64

Both methods share the same set of functions and properties, but in the case of direct link, all the names are preceded by 'dtmr' to avoid namespace collisions. For instance, the ActiveX function LastLtcFrame(), would be dtmrLastLtcFrame() in the direct link version.

ActiveX Usage (Deprecated)

Deprecated The ActiveX will install with the required DLLs into "C:\Program Files\MediaReactor" or "C:\Program Files(x86)\MediaReactor". The ActiveX component can be added to your project by inserting the MediaReactor ActiveX control via your IDE. For the functions below, remove the dtmrXXX start.

Direct Link Usage

All the functions in the direct link model have 'dtmr' prepended to the function name. This means the 'GetVideoFrame' becomes 'dtmrGetVideoFrame' to avoid naming conflicts. The direct link setup depends on the platform being used:

Windows 32
"C:\Program Files\MediaReactor"
Windows 64 – using 32 bit
"C:\Program Files(x86)\MediaReactor"
Windows 64 – using 64 bit
"C:\Program Files\MediaReactor"

macOS
/Libraries/Frameworks/DrasticDDR.framework

Linux 64
/usr/bin
/usr/lib

To use the direct link, you will need to include “dtmediaread.h” in your source file, and link to libdtmediaread.lib/.a/framework, depending on your platform.

Soft link is also an option for the direct link API. Each function prototype includes a function pointer typedef. It is the same as the prototype with a 'p_' added to the front. The SDK also ships with a C file dtmr_loader.cpp that has all the functions as point, and a load/unloader function for your convenience.

Methods and Properties

dtmrOpen

long dtmrOpen(BSTR bstrFileName, long dwFlags);

Open a new file, stream, or network source for reading. The BSTR is a UTF-8 string on all platforms (Windows will be automatically converted to Unicode). This operation will attempt to open as much media as is available automatically. Along with the specified file, any 'side bar' audio file will be added, as well as information files (XML, RDF, TCI, NDX) and time code tracks. In the case of a series of stills, only one still needs to be specified. DTMediaRead attempts to treat everything as a video stream, so if it can build a sequence out of stills, it will. Here are the basic file types that may be automatically added to your main video stream:

xxx.avi	- Main stream (could be .mov, .gen, .omf, etc.)
xxx.wav	- Will replace audio 1 & 2
xxx.A12.wav/aiff	- Also, will replace audio 1 & 2
xxx.AA.wav/aiff	- Also, will replace audio 1 & 2
xxx.A1.wav/aiff	- Mono wave pairs
xxx.A34.wav	- Add more audio channels
xxx.XML	- Metadata information (dt: space)
xxx.RDF	- Uniform description XML
xxx.TCI	- Time code information file
xxx.NDX	- Frame index file
xxx.TC	- Time code stream

dtmrOpenMulti

DTMRHANDLE dtmrOpenMulti(char * szFileNameVAA[17], unsigned long dwFlags);

Similar to dtmrOpen, but used to open a group of one (or no) video file, and up to 16 audio files. Pointers are const, and must be filled in order except for video, which may be null.

dtmrClose

long dtmrClose(DTMRHANDLE dtmr);

Close the currently open opaque handle to the stream or file.

dtmrSourceFileName

void dtmrSourceFileName(DTMRHANDLE dtmr, char * tszMAX_PATHString);

The final file name used for the source file returned as UTF-8.

dtmrSourceHeight

long dtmrSourceHeight(DTMRHANDLE dtmr, long *pVal);

Source video media's height.

dtmrSourceWidth

long dtmrSourceWidth(DTMRHANDLE dtmr, long *pVal);

Source video media's width.

dtmrSourceBitDepth

long dtmrSourceBitDepth(DTMRHANDLE dtmr, long *pVal);

Source video media's bit depth.

dtmrSourceFourCC

long dtmrSourceFourCC(DTMRHANDLE dtmr, long *pVal);

Source video media's four character code compression type.

dtmrSourceBitRate

long dtmrSourceBitRate(DTMRHANDLE dtmr, long *pVal);

Source video media's bit rate.

dtmrSourceFrameSize

long dtmrSourceFrameSize(DTMRHANDLE dtmr, long dwFrame, long *pVal);

Source video media's frame size for the requested or current frame. This will return the minimum size required for the frame, including padding, and should be used to pass into the read, unless a different padding is required.

dtmrSourceVideoChannels

long dtmrSourceVideoChannels(DTMRHANDLE dtmr, long *pVal);

Source video total channels as a bitwise array.

dtmrSourceAudioChannels

long dtmrSourceAudioChannels(DTMRHANDLE dtmr, long *pVal);

Source audio total channels as a bitwise array. Here are some examples of the dtmrSourceAudioChannels return:

Mono = 0x0001 (1)
Stereo = 0x0003 (2)
Quad = 0x000F (15)
Eight Channel = 0x00FF (255)

dtmrSourceAudioFrequency

long dtmrSourceAudioFrequency(DTMRHANDLE dtmr, long *pVal);

Source audio media frequency.

dtmrSourceAudioBitsPerSample

long dtmrSourceAudioBitsPerSample(DTMRHANDLE dtmr, long *pVal);

Source audio media bits per sample.

dtmrSourceAudioFourCC

long dtmrSourceAudioFourCC(DTMRHANDLE dtmr, long *pVal);

Source audio four character code. The most common would be 0/1 – little endian PCM and 'sowt' – big endian PCM.

dtmrDuration

long dtmrDuration(DTMRHANDLE dtmr, long *pVal);

Return the duration (total number of frames) of the media.

dtmrAudioDuration

long dtmrAudioDuration(DTMRHANDLE dtmr, long *pVal);

Return the audio duration (total number of samples) of the media.

dtmrSourceRate

long dtmrSourceRate(DTMRHANDLE dtmr, long *pVal);

Source video rate value (FPS = SourceRate / SourceScale)

dtmrSourceScale

long dtmrSourceScale(DTMRHANDLE dtmr, long *pVal);

Source video scale value (FPS = SourceRate / SourceScale)

> NOTE: It is best to use standard Rate/Scale descriptors when setting up files. Here are the most common: 24/1, 24000/1001, 25/1, 30000/1001, 30/1, 50/1, 60000/1001, 60/1

dtmrSourceMetaDataDWORD

long dtmrSourceMetaDataDWORD(DTMRHANDLE dtmr, long dwMetaDataElement, long *pVal);

Return source metadata information that are numeric (DWORDs or longs). See the **Metadata Elements** section towards the end of the manual. Works for vvwiTimeCode to vvwiWhiteBalance inclusive, and vvwiVideoWidth to vvwiAudioBits inclusive.

dtmrSourceMetaDataSTR

BSTR * dtmrSourceMetaDataSTR(DTMRHANDLE dtmr, long dwMetaDataElement, char * szMAX_PATHString);

Return source metadata information that are string data. See the **Metadata Elements** section towards the end of the manual. Works for vvwiFileName to vvwiUMID inclusive.

dtmrGetReadTypes

long dtmrGetReadTypes(DTMRHANDLE dtmr, unsigned long dwIndex, unsigned long * pdwTypes);

Returns recommended and supported read types. Please see the **Output Video Formats** section for more information on the available types. One type is returned for each index specified until DTMR_READTYPE_INVALID is returned. The first return (where dwIndex = 0) will always be DTMR_READTYPE_RGBA as all video types can decode to our native RGBA. The next return depends on the source type. For RGB(A) sources, dwIndex = 1 will return DTMR_READTYPE_INVALID indicating that only RGBA decoding is supported. Below are a few sample returns for different file types. dtmrGetReadTypes() should always be checked unless you are using dtmrSetReadType(DTMR_READTYPE_RGBA).

Type	dwIndex	Return
DPX:	0	DTMR_READTYPE_RGB10Bit
	1	DTMR_READTYPE_RGBA
	2	DTMR_READTYPE_INVALID
v210Mov:	0	DTMR_READTYPE_V210
	1	DTMR_READTYPE_UYVY
	2	DTMR_READTYPE_RGBA
	3	DTMR_READTYPE_INVALID
AbacusYUV:	0	DTMR_READTYPE_UYVY
	1	DTMR_READTYPE_RGBA
	2	DTMR_READTYPE_INVALID
MPEG:	0	DTMR_READTYPE_UYVY
	1	DTMR_READTYPE_RGB
	2	DTMR_READTYPE_INVALID
DNG:	0	DTMR_READTYPE_RGBHALFFLOAT
	1	DTMR_READTYPE_RGB48
	2	DTMR_READTYPE_RGBA
	3	DTMR_READTYPE_INVALID

(Note: Formats such as MPEG, MJPEG and MPEG-4 will return DTMR_READTYPE_UYVY as a possible type because they are YCbCr based. The DTMR_READTYPE_UYVY frame will always return 4:2:2 YCbCr interleaved samples even if the source format is a lower sampling rate such as 4:2:0 or 4:1:1)

dtmrSetReadType

long dtmrSetReadType(DTMRHANDLE dtmr, long lReadType);

Set the read type for the video frames. Please see the **Output Video Formats** section for more information on the available types. This function should only be set to one of the types available as specified in the GetReadTypes() return.

Set read type also allows the caller to set the audio bit size returned for 16 bit or 32 bit. All audio will be decoded and converted to the requested format. If this is not set, it will return 16 for 16 and 32 for any other bit depth.

dtmrGetFrame

dtmrSetFrame

long dtmrGetFrame(DTMRHANDLE dtmr, long *pVal);
long dtmrSetFrame(DTMRHANDLE dtmr, long newVal);

Get or set the current absolute (zero based) frame.

dtmrSetVideoChannel

long dtmrSetVideoChannel(DTMRHANDLE dtmr, long lVideoChannel);

Set the video channel to be read by get video frame.

dtmrGetVideoFrame

long dtmrGetVideoFrame(DTMRHANDLE dtmr, unsigned char * psvFrame, long * plSize);

GetVideoFrame returns a safe array containing one video frame. Passing NULL psvFrame will return size of the allocation for the frame, including any temporary space. Once allocated, the plSize should include the actual frame size. This can be retrieved from dtmrSourceFrameSize() for the smallest representation. Larger values can be passed in, and this will cause the read to pad out to those sizes, if possible. This is only available for RGB frame types. Please note, this function will only return frames in a few specified formats. These formats do not change regardless of the parameters returned by the SourceXXX methods. Please see the **Output Video Formats** section for more information on the available types.

dtmrAudioChannelPair

long dtmrSetAudioChannelPair(DTMRHANDLE dtmr, long lAudioChannelPair);

Set the audio channel pair to be read by get audio frame. Since the audio frame returns a stereo set, you can then select the two channels you want to read from the available bits. Basically:

0 - 0x0003
1 - 0x000C
2 - 0x0030
3 - 0x00C0

dtmrGetAudioFrame

long dtmrGetAudioFrame(DTMRHANDLE dtmr, unsigned char *

psaFrame, long * plSize);

GetAudioFrame returns a safe array containing one video frame worth of audio data. Please note that the audio data is always returned as uncompressed, stereo PCM regardless of the values describing the source material's type returned by SourceXXX methods. Please see the **Output Audio Formats** section for more information.

dtmrGetCurExtendedData

dtmrGetCurCloseCaptions

long dtmrGetCurExtendedData(DTMRHANDLE dtmr, unsigned char *pvData, unsigned long * plFlags, long *plSize);
long dtmrGetCurClosedCaptions(DTMRHANDLE dtmr, unsigned char *pvCC, long *plCCSize, long * plCCFlags);

Get current extended data or closed captions (*NOTE: to set this for a particular frame, dtmrSetFrame and dtmrGetVideoFrame() or dtmrGetFileFrameInfo() must be called first*). Normally both these calls return some combination of closed captions. With certain files, like Navy or NASA embedded data, the extended data may be something other than closed captions. The first two bytes are always CC1/CC3. If the FRAMEINFO_DATA_F1_EIA608 flag is not set, their value is undefined, but will likely be 0x80 0x80. The second two bytes are always CC2/CC4 if the FRAMEINFO_DATA_F2_EIA608 flag is set, otherwise they are undefined but will likely be 0x80 0x80. Everything from byte 4 on are 708 or OP-47 SMPTE 436 packets of closed captions, active format description and V-Chip IDs. Each ANC packet will start with its DID SDID and size (for example for 708 captions 0x61 0x01 0x49). That size can be used to run through multiple ANC packets for a given frame. The CC, if it exists, will always be first, followed by any AFD, V-Chip or other custom packets.

```
//! Data is EIA-608B SD closed caption data field one (uses 2 bytes)
#define FRAMEINFO_DATA_F1_EIA608                    0x00000001
//! Data is EIA-608B SD closed caption data field two (uses 2 bytes)
#define FRAMEINFO_DATA_F2_EIA608                    0x00000002
//! Data is EIA-708 HD closed caption data (uses remaining bytes = minus the above)
#define FRAMEINFO_DATA_EIA708                  0x00001000
//! Data is OP-47 closed caption data
#define FRAMEINFO_DATA_OP47                         0x00002000
```

dtmrSourceAudioFourCC

long dtmrSourceAudioFourCC(DTMRHANDLE dtmr, long *pVal)

Get the audio four character code

dtmrSetVideoChannel

long dtmrSetVideoChannel(DTMRHANDLE dtmr, long lVideoChannel);

Select the video channel to read. This is a bitwise array.

dtmrAudioChannelPair

long dtmrSetAudioChannelPair(DTMRHANDLE dtmr, long lAudioChannelPair);

Select the audio pair to read. This is NOT bitwise. Each pair is a number starting at 0:

Pair	Bitwise Channel Set
0	0x0003
1	0x000F
2	0x0030
3	0x00F0

dtmrSetMode

long dtmrSetMode(DTMRHANDLE dtmr, void * pMediCmd);

Set mediacmd (advanced). This can be used to adjust advanced settings like GPU enable, number of threads in codec, number of threads in file, white balance, matrix enable, and other settings. Please contact Drastic for more information.

dtmrLastVitcFrame

long dtmrLastVitcFrame(DTMRHANDLE dtmr, long *pVal);

Return the last dtmrGetVideoFrame VITC (vertical blank) time code as a frame value. To set this for a particular frame, dtmrSetFrame and dtmrGetVideoFrame() or dtmrGetFileFrameInfo() must be called first.

dtmrLastVitcUb

long dtmrLastVitcUb(DTMRHANDLE dtmr, long *pVal);

Return the last GetVideoFrame VITC (vertical blank time code) user bits. To set this for a particular frame, dtmrSetFrame and dtmrGetVideoFrame() or dtmrGetFileFrameInfo() must be called first.

dtmrLastLtcFrame

long dtmrLastLtcFrame(DTMRHANDLE dtmr, long *pVal);

Return the last dtmrGetVideoFrame LTC (SMPTE) time code. To set this for a particular frame, dtmrSetFrame and dtmrGetVideoFrame() or dtmrGetFileFrameInfo() must be called first.

dtmrLastLtcUb

long dtmrLastLtcUb(DTMRHANDLE dtmr, long *pVal);

Return the last dtmrGetVideoFrame LTC (SMPTE time code) user bits. To set this for a particular frame, dtmrSetFrame and dtmrGetVideoFrame() or dtmrGetFileFrameInfo() must be called first.

dtmrGetFileFrameInfo

long dtmrGetFileFrameInfo(DTMRHANDLE dtmr, unsigned long dwFrame, unsigned long dwChannels, unsigned long dwFlags, size_t * pnPosition, size_t * pnSize, unsigned long * pdwFrameFlags, char * szFilePathAndName);

This call returns information about a frame (or group of samples) of audio or video. It will return the position, size, frame flags, and file name for a video sample or audio sample groups.

```
        //! Send this in if you just need the filename (faster than getting all
the info)
#define DPOSSIZENAME_FILENAME_ONLY          0x40000000        //
Same as DFRAME_SKIP_FRAME
        //! Flag for mediafile/avhal to get audio dframe
#define GetAudio   0x00000000
        //! Flag for mediafile/avhal to get video dframe
#define GetVideo   0x00000001

// dwFrameFlags
#define DPOSSIZENAME_VIDEO_FRAME            0x00000001
        //! Is this file type currently recording
#define DPOSSIZENAME_RECORDING              0x00000004
        //! This frame needs to be made black (default frame) in MediaFile
#define DPOSSIZENAME_PLEASE_BLACK
_PDFRAMEFLAGS_PLEASE_BLACK      //      0x00000080
        //! This is a mono audio chunk
#define DPOSSIZENAME_MONO_AUDIO_FRAME     0x00000100
        //! This is a stereo audio chunk
#define DPOSSIZENAME_STEREO_AUDIO_FRAME   0x00000200
#define DPOSSIZENAME_QUAD_AUDIO_FRAME     0x00000400
#define DPOSSIZENAME_4_1_AUDIO_FRAME      0x00000800
#define DPOSSIZENAME_5_1_AUDIO_FRAME      0x00001000
#define DPOSSIZENAME_7_1_AUDIO_FRAME      0x00002000
```

```
#define DPOSSIZENAME_9_1_AUDIO_FRAME        0x00004000
#define DPOSSIZENAME_AUDIO_MASK
(DPOSSIZENAME_MONO_AUDIO_FRAME|
DPOSSIZENAME_STEREO_AUDIO_FRAME|
DPOSSIZENAME_STEREO_AUDIO_FRAME|
DPOSSIZENAME_QUAD_AUDIO_FRAME|
DPOSSIZENAME_4_1_AUDIO_FRAME|DPOSSIZENAME_5_1_AUDIO_FRAME|
DPOSSIZENAME_7_1_AUDIO_FRAME|DPOSSIZENAME_9_1_AUDIO_FRAME)
#define DPOSSIZENAME_FRAME_MASK                    0x0000FFFF
      //! This frame contains audio data see DFRAME::dwType
#define DFRAME_TYPE_AUDIO          0x00010000
      //! 16 bit audio
#define DPOSSIZENAME_AUD_16_16_BIT          0x00100000
      //! 20 bit audio in 24
#define DPOSSIZENAME_AUD_20_24_BIT          0x00200000
      //! 24 bit audio in 24
#define DPOSSIZENAME_AUD_24_24_BIT          0x00400000
      //! 24/32 bit audio in 32
#define DPOSSIZENAME_AUD_24_32_BIT          0x00800000
      //! 32/32 bit audio in 32
#define DPOSSIZENAME_AUD_32_32_BIT          0x01000000
      //! Audio is compressed
#define DPOSSIZENAME_AUD_COMPRESSED              0x02000000
      //! Audio is big endian, else little endian
#define DPOSSIZENAME_AUD_BIGENDIAN_BIT      0x00080000
      //! Just for completeness
#define DPOSSIZENAME_AUD_LITTLEENDIAN_BIT 0x00000000
      //! This frame is independent of other frames for decode see
DFRAME::dwType
#define DFRAME_TYPE_KEYFRAME      0x10000000
      //! This frame is independent of other frames for decode (an MPEG I
Frame) see DFRAME::dwType
#define DFRAME_TYPE_KEYFRAME_I   0x10000000
      //! This frame requires previous keyframe(s) (for MPEG a P Frame) see
DFRAME::dwType
#define DFRAME_TYPE_KEYFRAME_P   0x80000000
      //! This frame requires more than one frame to decode (for MPEG a B
Frame) see DFRAME::dwType
#define DFRAME_TYPE_KEYFRAME_B   0x20000000
      //! This frame should be skipped (decoded, but not displayed) - Used
to reach seek frame on a non key frame from key frame see
DFRAME::dwType
#define DFRAME_SKIP_FRAME          0x40000000
```

dtmrGetCaptureDiscontinuities

long dtmrGetCaptureDiscontinuities(DTMRHANDLE dtmr, unsigned char ** pszDiscontinuities);

Return the list, or part of the list, of discontinuities that have occurred while capturing the file. The function may need to be called more than once, for long lists of discontinuities. The format is XML, with each individual <Discontinuity ...> is with a single <DISCONTINUITIES> </DISCONTINUITIES> key pair. The first call will return nothing, if there are no discontinuities. If there are, the first call will start with <DISCONTINUITIES> followed by lines of <Discontinuity .. >. For small numbers the entire list will be returned on the first call with the ending </DISCONTINUITIES>. For larger lists, subsequent calls will return the next set of discontinuity lines, until all are returned along with the closing </DISCONTINUITIES> tag. In the case that the file you're checking is still writing you can treat it the same as larger lists, just keep calling dtmrGetCaptureDiscontinuities and you'll get the next piece if available.

If you want force a reread just free the pointer with dtmrFreeCaptureDiscontinuities, and the next call to dtmrGetCaptureDiscontinuities will start from the beginning.

Below is sample list:

<DISCONTINUITIES>
<Discontinuity type="video" time="10:25:24 Wednesday, July 27, 2016" frame="286" timecode="2046723" timecodetype="D">18:58:12;28</Discontinuity>
<Discontinuity type="audio" time="10:25:24 Wednesday, July 27, 2016" sample="13728000" timecode="2046723" timecodetype="D">18:58:12;29</Discontinuity>
<Discontinuity type="meta tc" time="10:25:24 Wednesday, July 27, 2016" frame="286" timecode="2046723" timecodetype="D">18:58:12;29</Discontinuity>
<Discontinuity type="timecode" time="10:25:25 Wednesday, July 27, 2016" frame="305" timecode="2046723" timecodetype="D" comment="Last I TC: 2046723 Current: 2046423">18:58:13;17</Discontinuity>
</DISCONTINUITIES>

dtmrFreeCaptureDiscontinuities

long dtmrFreeCaptureDiscontinuities(DTMRHANDLE dtmr, unsigned char ** pszDiscontinuities);

Free the buffer returned by dtmrGetCaptureDiscontinuities. This only needs to be called after all the calls to the 'get' function, and to the entire list have been returned.

Clip Detection API

DTMediaRead includes a clip detection API. This API finds clips within a directory structure, like P2 or BDMV, as well as sequences of files and other standard formats. It also includes stitching together of files that were recorded as segments, and combining audio, video, closed caption and metadata files as a single entity. Once found, the clips may be opened with the read API above, and the individual elements can be retrieved, as well as reading each clip with all its components automatically.

Header: detectdir.h

ddtFileDir
Normal file directory

ddtAVCHD
Sony, JVC, Panasonic, Canon camera format
AVCHD
AVF_INFO
private
AVCHD
BDMV
CLIPINF
PLAYLIST
STREAM
SONY

ddtBDMV
BDMV - BluRay, subset of AVCHD
BDMV
CLIPINF
PLAYLIST
STREAM

ddtXAVCS
Sony XAVC-S cameras
XAVC-S
M4ROOT
CLIP
GENERAL
SUB
THMBNL

ddtP2Atom
Panasonic P2 Atom (DV25, 50, 100, AVCi 100)
CONTENTS
  AUDIO
  CLIP
  ICON
  PROXY
  VIDEO
  VOICE

ddtP2OP1b
Panasonic P2 OP1b (AVCi 100/200, Long G)
CONTENTS
  AUDIO
  AVCLIP
  CLIP
  ICON
  PROXY
  VIDEO
  VOICE

ddtXDCam
Sony XDCam
 Clip
 Edit
 General
 Sub

ddtHDV
Sony HDV Camcorder

ddtC500
Canon C Series 500/700

ddtCanonXF
Canon XF Cameras
 CONTENTS
  CLIPS001
   <NAME>
   JOURNAL

ddtGVJpeg2000
Grass Valley
 <Name>.bmp
 <Name>.xml
 <Name>_DM.xml
 <Name>-<num>_H.mxf
 <Name>-<num>_H.mxf

ddtCodex
Codex RAW formats
 <FileNameAsDir>
   <FileName><NUM>.rmf
 <FileNameBase>.xml

ddtGemini
Gemini Recorder
 <FileNameAsDir>
   <FileNameRoot>.xml
   <FileName><NUM>.rmf

ddtGF
GF (Similar to P2)
 BIN001
   ANC
   AUDIO
   CLIPINF
   PIC
   PROXY
   VBI
   VIDEO
 PLAYLIST

ddtArriRaw
ARRI Raw

ddtRed
RED camera

ddtHasSequences
Generic sequence(s)

ddtK2Server
Grass Valley server (K2)

audio.A#
data.N#
timecode.T#
video.F#
<clipname>.xml

detectDirType __stdcall ddtCheckDir(char * szDirectory);

Quickly determine the type of the directory specified

DDTHANDLE __stdcall ddtOpenDir(char * szDirectory, detectDirType ddType);

Open a directory as a particular type

DWORD __stdcall ddtCloseDir(DDTHANDLE ddtH);

Close a previously open directory

DWORD __stdcall ddtGetTotalFiles(DDTHANDLE ddtH);

Get total files/clips

DWORD __stdcall ddtGetFilePath(DDTHANDLE ddtH, int nIndex, char * szFullPath);

Get a clip file name with full path

DWORD __stdcall ddtGetBaseName(char * szFullPath, char * szBaseName);

Get base file name for sequences

DWORD __stdcall ddtGetFileInfo(DDTHANDLE ddtH, int nIndex, WIN32_FIND_DATA * pFileInfo);

Get a clip's file info

DWORD __stdcall ddtCheckSequence(char * szFileName);

Detect if file is part of a sequence

Defines And Constants

These formats are used by dtmrGetReadTypes() and dtmrSetReadType() to set up the frame return type for dtmrGetVideoFrame(). See the **Video Output Formats** section for more information on these frame layouts.

//! Windows RGBA (like bitmap, tga, etc.)
const unsigned long **DTMR_READTYPE_ARGB** = 0;
//! RGB 8 bits per component, 24 total
const unsigned long **DTMR_READTYPE_RGB** = 0x10000000;
//! Alpha only 8 bits per component, repeated to 24
const unsigned long **DTMR_READTYPE_AAA** = 0x20000000;
//! 8 Bit YCbCr (yuv2, D1/HDSDI raw 4:2:2 video
const unsigned long **DTMR_READTYPE_UYVY** = 1;
//! 10 Bit v210 (quicktime packing) 4:2:2 video
const unsigned long **DTMR_READTYPE_V210** = 2;
//! 10 Bit RGB 4:4;4 (dpx packing)
const unsigned long **DTMR_READTYPE_RGB10Bit** = 3;
//! 16 bit per component (64 bit) RGBA 4:4:4:4
const unsigned long D**TMR_READTYPE_RGBA64** = 4;
//! RGB 16 bits per component, 48 total
const unsigned long **DTMR_READTYPE_RGB48** = 0x10000004;
//! Alpha only 16 bits per component, repeated to 48
const unsigned long **DTMR_READTYPE_AAA16** = 0x20000004;
//! 16 bit half float per component RGBA (GPU)
const unsigned long **DTMR_READTYPE_RGBAHALFFLOAT** = 5;
//! 16 bit half float per component RGB (GPU)
const unsigned long **DTMR_READTYPE_RGBHALFFLOAT** = 6;
//! Set to invert the picture vertically
const unsigned long **DTMR_READFLAG_FLIP** = 0x80000000;
//! Invalid file
const unsigned long **DTMR_READTYPE_INVALID** = -1;

Output Video Formats

These are the formats supported by GetVideoFrame(). Each of these formats only appears as specified here for this return. The SourceXXX series of methods (including SourceBitDepth and SourceFourCC) refer to the video media as it is saved on disk. The DTMediaRead library will decompress, and where necessary convert, from the file's native format to the requested format set by SetReadType(). For each file opened, the GetReadTypes() should be called to determine the available read types.

ARGB 32 (8 bits per component, vertical invert)

DTMR_READTYPE_RGBA

ARGB Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
Alpha								Red								Green								Blue							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

RGB 24 (8 bits per component, vertical invert)

DTMR_READTYPE_RGB

RGB Decreasing Address Order																															
								Byte 2								Byte 1								Byte 0							
								Red								Green								Blue							
								7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

AAA 24 (8 bits per component, vertical invert)

DTMR_READTYPE_AAA

RGB Decreasing Address Order																															
								Byte 2								Byte 1								Byte 0							
								Alpha								Alpha								Alpha							
								7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

RGB 30 (10 bits per component)

DTMR_READTYPE_RGB10Bit

RGB 10 Bit Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
Blue								Green				Blue				Red		Green						Red							
5	4	3	2	1	0			3	2	1	0	9	8	7	6	1	0	9	8	7	6	5	4	9	8	7	6	5	4	3	2

Please note: This is the standard DPX file layout, which was originally

big endian, but is viewed here as little endian.

YCrCb 8 (8 bits per component 4:2:2)

DTMR_READTYPE_UVYV

YCbCr8 2 Pixels, Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
Cr								Y1								Cb								Y0							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

YCrCb 10 (10 bits per component 4:2:2)

DTMR_READTYPE_V210

YCbCr10 Pixels, Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
		Cr 0								Y 0										Cb 0											
		9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
Byte 7								Byte 6								Byte 5								Byte 4							
		Y 2								Cb 1										Y 1											
		9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
Byte 11								Byte 10								Byte 9								Byte 8							
		Cb 2								Y 3										Cr 1											
		9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
Byte 15								Byte 14								Byte 13								Byte 12							
		Y 5								Cr 2										Y 4											
		9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0

RGBA 4:4:4:4 16 Per (64 Total) Integer

//! 16 bit per component (64 bit) RGBA 4:4:4:4
const unsigned long DTMR_READTYPE_RGBA64 = 4;

RGB 4:4:4 16 Per (48 Total) Integer

//! RGB 16 bits per component, 48 total
const unsigned long DTMR_READTYPE_RGB48 = 0x10000004;

Alpha 16 Bit Integer (48 Total Repeated 3 Times)

//! Alpha only 16 bits per component, repeated to 48

```
const unsigned long DTMR_READTYPE_AAA16 = 0x20000004;
```

RGBA 4:4:4:4 16 Per (64 Total) Half Float

```
//! 16 bit half float per component RGBA (GPU)
const unsigned long DTMR_READTYPE_RGBAHALFFLOAT = 5;
```

RGB 4:4:4 16 Per (48 Total) Half Float

```
//! 16 bit half float per component RGB (GPU)
const unsigned long DTMR_READTYPE_RGBHALFFLOAT = 6;
```

Image Invert

```
//! Set to invert the picture vertically
const unsigned long DTMR_READFLAG_FLIP = 0x80000000;

//! Invalid file
const unsigned long DTMR_READTYPE_INVALID = -1;
```

Supported Video IP Types

UDP and RTP

UDP [User Datagram Protocol] and RTP [Real-time Transport Protocol] streams can be elementary video or audio streams, or more commonly a transport stream with PMT/PAT (Program Association Table/Program Mapping Table) and a number of streams within it. For UDP and RTP, you can specify a TCP (direct) address, but normally it will be a multicast group address, and also a port is normally specified. Here are a few examples:

udp://239.254.40.40:5004
rtp://239.100.20.20:50004
rtp://239.100.30:31:1234

This is a server protocol on the receiver, and requires the selected port to be open to receive. On the send side, it should work without firewall adjustment.

SRT

SRT [Secure Reliable Transport] streams contain a transport stream with PMT/PAT and a number of streams within it. For SRT you can specify an address and a port. There are three modes for SRT: listener, caller and rendezvous. If you are a listener, you can only connect with a caller and vice versa. For Rendezvous, both the sender and receiver must be in rendezvous mode. A password for encrypted service can also be set. Here is some information on the modes:

listener - this has to be one of your local IP addresses, and acts as a server waiting for a connection, so it must be directly visible to the caller (not behind a firewall)
caller - this calls out to a remote IP that is running as a listener. You must be able to reach the IP directly (e.g. no firewall)
rendezvous - this connects bi directionally, allowing it to connect through firewalls without extra configuration. Each side of the rendezvous uses the external (internet facing) IP address of their internet connection. This allows the signals to connect and pass through the firewall

Here are a few examples:

srt://239.254.40.40:5004?mode=listener

srt://172.12.25.20:5006?mode=caller
srt://239.100.30:31:1234?mode=caller&password=thisisapassword&user=thisisauser

Possible parameters include

mode=
caller
listener
rendezvous
password=<string>
keylen=16|24|32
username=<string>
streamid=#
latency=#
buffering=#
maxbw=#

When using the 'listener' mode, the port it is listening on must be open in the firewall. For Caller and Rendezvous, it should work without firewall adjustment.

RIST

RIST [Reliable Internet Stream Transport] streams are UDP based self correcting connections. Drastic currently supports the following RIST profiles: Simple, and Main. Simple Profile provides ARQ (automatic repeat request) for packet loss recovery, jitter removal, optional FEC (forward error correction). The Main Profile adds encryption for secure content.

Both the sender and the receiver must be in the same mode. The receiver will be the server and listen for a connection. The sender will be the client and connect to the receiver to send the data. The protocol will use two ports, the lower of which is specified in the URL and the higher which is the lower plus one. The lower port must be even.

Here are a few examples:

rist://10.0.0.123:5000?mode=listener&profile=main
rist://192.168.1.22?mode=caller&profile=simple

Possible parameters include:

mode: listener (for server/receiver), caller (for client/sender) - Required
profile: simple. main or advanced
password: encryption key
buffering: amount of buffer in milliseconds

When using the 'listener' mode, the port it is listening on must be open in the firewall. For Caller, it should work without firewall adjustment.

RTSP

RTSP [Real Time Streaming Protocol] streams require not only the device address, but also the description of the source of the stream you are accessing on that device. RTSP are also often user/password protected, so you may have to send a user/password in the form "<user>:<pass>@" just before the device identifier. Here are a few examples, and their sources:

rtsp://192.168.100.10/axis-media/media.amp (an Axis camera)
rtsp://192.168.199.11/user:pass@/video1+audio1 (a Marshall camera, with password)
rtsp://192.168.160.20:/onvif/media.amp (an OnVIF source)
rtps://192.168.150:11/video1?videocodec=h264 (a Marshall camera, video only, force h.264)

For sending, RTSP should work without firewall adjustment. RTSP uses port 554

RTMP

RTMP [Real-Time Messaging Protocol] is normally used to stream one video and one stereo audio channel to a website for distribution to multiple watchers. In modern sites, the RTMP is actually re-wrapped into HLS, which is then viewed by the end user. To connect to an RTMP site, like flowcaster.live, youtube.com, and twitch.com, you will need the URL/Link and the key/secret. For youtube, they are available after you 'go live' as the Stream URL and the Stream Key. Once you have them, you simply add a slash and the Stream Key to the Stream URL.. For example:

Stream URL: rtmp://a.rtmp.youtube.com/live2

Stream Key: j2bg-a6ck-8t48-w2y2-aaaa

Final URL: rtmp://a.rtmp.youtube.com/live2/j2bg-a6ck-8t48-w2y2-aaaa

For sending, RTMP should work without firewall adjustment. RTMP uses port 1935

WebRTC

WebRTC [Web Real-Time Communication] is a browser native method of sharing video, audio and data. It is primarily used in chat programs, like Google Meet. When sending via WebRTC, FlowCaster appears as a person in the chat, with whatever video and audio it is receiving being sent to the chat.

Here is an example:

webrtc://flowcaster.live?meetingid=asre-dsec-asds-seff&name=flowcaster

WebRTC uses a bunch of standard ports:

Access to ports TCP + UDP 4443, 3478, 443 for www.flowcaster.live
Access to video streaming services in VPN and Firewall settings
Ports used: 80, 443, 4443, 3478 (TCP and UDP), 5349 TCP, 40000:65535 UDP

WHIP (WebRTC - Millicast)

WHIP [WebRTC-HTTP ingestion protocol] is a simpler negotiation system for WebRTC. Currently in use by Millicast to receive streams for worldwide, low latency transmission, FlowCaster and Net-X-Code Server support sending video signals via WHIP. WHIP requires an auth code (available from the Millicast config pages) and a stream name. The stream name is added to the end of whip://director.millicast.com/api/whip/ and the auth token is a parameter that starts with auth=.

Here is an example
whip://director.millicast.com/api/whip/kwky3g6g?
auth=48ce3daa09cd8355f80fc0d37005f9422a62bebf9b6411b61cfb1cfb2fa

WHIP uses a bunch of standard ports:

Access to ports TCP + UDP 4443, 3478, 443 for www.flowcaster.live
Access to video streaming services in VPN and Firewall settings

Ports used: 80, 443, 4443, 3478 (TCP and UDP), 5349 TCP, 40000:65535 UDP

BLS (Bliss Protocol)

BLS [Browser Live Stream] is a protocol developed by Drastic to send live video via an encrypted channel directly to a user's browser. It allows for much higher quality video than WebRTC, while still not requiring any plugins or special setup to present audio and video directly in a modern, HTML5 browser.

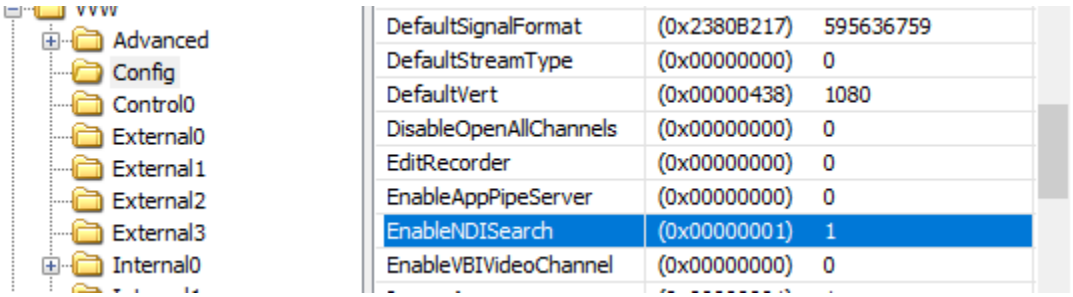
Here are a couple examples:

bls://10.0.0.234:5000
blss://192.168.202.200:3000?password=kfiwgt84jsd&remoteip=120.32.54.6

BLS uses the port explicitly set. If there is no port set, it will use 80 for unencrypted and 443 for encrypted traffic.

NDI

NDI [Network Device Interface] is a video over IP protocol from NewTek®. It requires a device name and a source name to access NDI sources. NDI sources may also be searched on the local network. To enable the search, run DDRConfig and select the Advanced tab. Go to /VVW/Config and change EnableNDISearch = 1. If it does not exist, then create a new Numeric value for it.



To specify an NDI stream, use the device name, followed by a space, and then the source name within brackets.

ndi://USER-PC (Desktop [2])
ndi://TestCameraSource (ISO_1)
ndi://PC2 (Google Chrome [1])

If you are creating an NDI stream, with FlowCaster or Net-X-Code Server, for instance, only the stream name is specified. The Computer name is added automatically by NDI, and you cannot use brackets in the name

ndi://FlowCasterOut
ndi://SDI1Out
ndi://SMPTE2110_Group1

NDI uses a range of TCP ports: NDI ports 49152 to 65535

CDI

CDI [Cloud Digital Interface] is an advanced, fully uncompressed, protocol for use within Amazon VMs. It transports video in a number of formats, as well as audio, time code and other metadata. While it is possible to use CDI with Amazon's enhanced network backbone, it is safest and most efficient, within their network stacks. The URL will include a local IP and port, with an optional remote IP, adapter and ID.

Here are some examples:

cdi://10.0.0.2:6000
cdi://10.0.0.1:6000?remoteip=10.0.0.200&adapter=EFA&id=2

Possible parameters include:

remoteip: a remote computer to connect to exclusively
adapter: the transport, EFA (Elastic Fabric Adapter) or socket. EFA is the default.
id: a numeric value to specify the stream

The implementation for this transit occurs over the Scalable Reliable Datagram (SRD) protocol. To achieve the highest performance and lowest latency, the AWS CDI SDK relies on EC2 instances that support the Elastic Fabric Adapter (EFA) and are placed within a single Placement Group.

The AWS CDI SDK opens one specified User Datagram Protocol (UDP) port per connection to control communication between Amazon EC2 instances running AWS CDI SDK. The receiving side listens on the specified port number. The transmitting side uses

a random port number from the ephemeral port range, as determined by the operating system.

For network security best practices concerning how to block UDP packets from the public Internet, see Security best practices for your VPC.

The AWS CDI SDK also relies on EC2 instances using a Security Group that allows all inbound and outbound traffic to and from the Security Group itself. For more information, see Prepare an EFA-Enabled Security Group.

S2022 and S2110

The SMPTE 2022-6 and SMPTE 2110 protocols can be accessed via SDP (Session Description Protocol) or manual setup. To access an SDP source:

s2202://192.168.101.200/channel1.sdp
s2110://mainsources.drastic.ca/crosspoint10.sdp

For some Drastic software, the source can be set up manually. For S2022, this is a single set of Source IP, Source Port, Destination IP, Destination Port and Interface address. One or any combination of these can be used to describe the source of the SMPTE 2022-6 stream, which contains all the video, audio and HANC/VANC channels. For SMPTE 2110, up to three sets of the same information are required to describe the video, audio and anc streams, which are all separate. A PTP (Precision Time Protocol) grandmaster may also be specified.

Output Audio Formats

These are the formats supported by dtmrGetAudioFrame(). By default, it will return video frame sized chunks of audio in a 16 bit container for 16 bit files, and 32 bit containers for other bit sizes. Using dtmrSetReadType, this can be modified to return a specific bit depth for the file, and switched from video frame audio sized groups to native/audio sample based reads:

```
//! Set readtype to video frame size AUDIO to 16 bits LE
const unsigned long DTMR_READTYPE_FRAME_AUDIO_16LE = (0x00010000 | 16);
//! Set readtype to video frame size AUDIO to 32 bits (note, 16, 20, 24 will be shifted to most significant, LE)
const unsigned long DTMR_READTYPE_FRAME_AUDIO_32LE = (0x00010000 | 32);
//! Set readtype to arbitrary sample AUDIO to 16 bits LE
const unsigned long DTMR_READTYPE_SAMPLE_AUDIO_16LE = (0x00110000 | 16);
//! Set readtype to arbitrary sample  AUDIO to 32 bits (note, 16, 20, 24 will be shifted to most significant, LE)
const unsigned long DTMR_READTYPE_SAMPLE_AUDIO_32LE = (0x00110000 | 32);
```

Audio is always output as two channels of either 32 bit or 16 bit per sample PCM audio. This is written in the same format as Windows wave files.

Left Channel (2 or 4 bytes little endian)
Right Channel (2 or 5 bytes little endian)
[repeats with no padding]

The frequency is dependent on the dtmrSourceAudioFrequency return. The bit size is dependent on dtmrSourceAudioBitsPerSample. If the dtmrSourceAudioBitsPerSample is 16 or less, then it will return 16 bit samples. If it is greater than 16 bits (normally 20, 24 or 32), then it will return 32 bits, where the 20 or 24 have been shifted up to become 32 bits.

The size of the return is dependent on the frame rate of the file. This can vary from 23.98 fps, or 2000/2001 samples per frame, down to 60 fps, or 800 samples per frame. The size will also vary, depending on how the frame rate divides into the sample rate. For example:

48,000 Hz audio at 29.97 video = 1601.6 samples
Because we can only return an even number of samples, the audio is returned in a 5 frame cadence of 1601 or 1602 samples. Because these are stereo, this means the application will receive 6404/6408 bytes in 16 bit, and 12808/12816 bytes in 20/24/32 bit.

Example - Direct

Follow these steps to read using the direct interface:

1. Open the file with dtmrOpen(szUTF8FileName, 0). Keep the opaque handle returned for all further calls.
2. Get the file information via the dtmrSourceHeight/Width/FourCC/etc. and the dtmrSourceAudioChannels/Frequency/etc.
3. Optionally, get the best read type from dtmrGetReadTypes
4. Set the read type you want for audio and video. For simplicity, start with dtmrSetReadType(h, DTMR_READTYPE_ARGB) and dtmrSetReadType(h, DTMR_READTYPE_FRAME_AUDIO_16LE).
5. Set the first frame you want to read dtmrSetFrame(h, 0)
6. Get the video suggested buffer size, and allocate memory for the video frame. Sending dtmrGetVideoFrame(h, NULL, &lSize) will return the suggested buffer size in lSize.
7. Read the video frame, passing in the actual video size of the frame. The actual size, without padding, can be retrieved from dtmrSourceFrameSize(). The second call to read would be dtmrGetVideoFrame(h, pAllocation, &lSourceFrameSize)
8. Read any audio channels you need with dtmrGetAudioFrame(h, pAudAllocation, &lAudioSize), which works the same way as video size.
9. Get any per frame metadata with dtmrLastVitcType/Frame/Ub, dtmrLastLtcType/Frame/Ub and dtmrGetCurExtendedData/dtmrGetCurClosedCaptions
10. Repeat steps 7 through 9 for any other frames you need
11. Close the opaque handle with dtmrClose(h)

Example – ActiveX [deprecated]

Follow these steps to load a media file into the DTMediaRead ActiveX SDK and then get the desired data.

1. Open a file. Somewhere in your application, you must open the desired file using the CDTReadX::Open(LPCTSTR bstrFileName, long dwFlags); function call. If the function returns -1, then the file has not been opened. Otherwise, proceed to step two.
2. Set the frame. After the file has been opened, set the current frame in the ActiveX by calling the CDTReadX::SetFrame(long nNewValue); where the nNewValue is the desired frame. This procedure can be attached to a slider control or an editbox, used for setting the frame dynamically. (NOTE: the total number of frames should be known before setting the current frame.)
3. Obtaining data. Once everything has been set up correctly, we can now access the ActiveX and retrieve the frame data for both audio and video. Variables you will need include a VARIANT to get the data, a DWORD to determine the size of the data, and a LPVOID to store the data.

   ```
   DWORD dwSize;
   VARIANT vid_frame;
   void* lpVidData;
   m_dtRead.GetVideoFrame(&vid_frame, (long*)&dwSize);
   ```

   If at this point, the size of the data (dwSize) is 0, then the ActiveX was unable to find any data for this frame. Now, access the data using the lpVidData variable. Continue if successful.

   ```
   if(SafeArrayAccessData(vid_frame.parray, &lpVidData) == S_OK)
   {
   ```

   Once we have access to the frame data, it can be manipulated however we like. It can be loaded into a device context, or saved to a bitmap, etc.
4. Freeing the data. Now that we are done with the data, it should be cleared.

   ```
   SafeArrayUnaccessData(vid_frame.parray);
   SafeArrayDestroy(vid_frame.parray);
   ```
5. To get the audio frame data from the ActiveX, perform step 3 using the CDTReadX::GetAudioFrame(VARIANT* psaFrame, long* plSize); function in place of GetVideoFrame.

Example: Reading Size/Position Via dtmrGetFileFrameInfo

Most file formats, and especially RTIndex files, can return the frame position and size, rather than the actual decoded audio/video data. This allows reading of files with your own decoders, and rewrapping of audio/video with direct reads to the data using normal OS reads. It is similar to a direct read, other than the read itself.

1. Open the file with dtmrOpen(szUTF8FileName, 0). Keep the opaque handle returned for all further calls.
2. Get the file information via the dtmrSourceHeight/Width/FourCC/etc. and the dtmrSourceAudioChannels/Frequency/etc.
3. Set the first frame you want to read dtmrSetFrame(h, 0)
4. Call dtmrGetFileFrameInfo with the frame, channel, and Audio/Video flags for the media you want. It will return the position, size, flags and file that media exists in. The flags will have audio info or video IPB flags.
5. Call dtmrGetFileFrameInfo for any other audio channels you want in the media file.
6. Get any per frame metadata with dtmrLastVitcType/Frame/Ub, dtmrLastLtcType/Frame/Ub and dtmrGetCurExtendedData/dtmrGetCurClosedCaptions
7. Repeat steps 3 through 6 for any other frames you need
8. Close the opaque handle with dtmrClose(h)

Metadata Elements

The functions dtmrSourceMetaDataDWORD() and dtmrSourceMetaDataSTR() use the defines below to return specific metadata from the file. The first enums are string values for dtmrSourceMetaDataSTR() (from vvwiFileName to vvwiUMID). The second set of enums are the DWORD values (from vvwiTimeCode to vvwiAudioBits).

```
/** Numeric values for all the metadata information types available in MR and VVW
*/
enum vvwInfoMetaTypes {
        //! see VVWINFO::szFileName
        vvwiFileName,
        //! see VVWINFO::szNativeLocator
        vvwiNativeLocator,
        //! see VVWINFO::szUniversalName
        vvwiUniversalName,
        //! see VVWINFO::szIP
        vvwiIP,
        //! see VVWINFO::szSourceLocator
        vvwiSourceLocator,

        //! see VVWINFO::szChannel
        vvwiChannel,
        //! see VVWINFO::szChannelName
        vvwiChannelName,
        //! see VVWINFO::szChannelDescription
        vvwiChannelDescription,
        //! see VVWINFO::szTitle
        vvwiTitle,
        //! see VVWINFO::szSubject
        vvwiSubject,
        //! see VVWINFO::szCategory
        vvwiCategory,                       // <-- 10
        //! see VVWINFO::szKeywords
        vvwiKeywords,
        //! see VVWINFO::szRatings
        vvwiRatings,
        //! see VVWINFO::szComments
        vvwiComments,
        //! see VVWINFO::szOwner
        vvwiOwner,
        //! see VVWINFO::szEditor
        vvwiEditor,
        //! see VVWINFO::szSupplier
        vvwiSupplier,
        //! see VVWINFO::szSource
        vvwiSource,
        //! see VVWINFO::szProject
        vvwiProject,
```

```
//! see VVWINFO::szStatus
vvwiStatus,
//! see VVWINFO::szAuthor
vvwiAuthor,                             // <-- 20
//! see VVWINFO::szRevisionNumber
vvwiRevisionNumber,
//! see VVWINFO::szProduced
vvwiProduced,
//! see VVWINFO::szAlbum
vvwiAlbum,
//! see VVWINFO::szArtist
vvwiArtist,
//! see VVWINFO::szComposer
vvwiComposer,
//! see VVWINFO::szCopyright
vvwiCopyright,
//! see VVWINFO::szCreationData
vvwiCreationData,
//! see VVWINFO::szDescription
vvwiDescription,
//! see VVWINFO::szDirector
vvwiDirector,
//! see VVWINFO::szDisclaimer
vvwiDisclaimer,                         // <-- 30
//! see VVWINFO::szEncodedBy
vvwiEncodedBy,
//! see VVWINFO::szFullName
vvwiFullName,
//! see VVWINFO::szGenre
vvwiGenre,
//! see VVWINFO::szHostComputer
vvwiHostComputer,
//! see VVWINFO::szInformation
vvwiInformation,
//! see VVWINFO::szMake
vvwiMake,
//! see VVWINFO::szModel
vvwiModel,
//! see VVWINFO::szOriginalArtist
vvwiOriginalArtist,
//! see VVWINFO::szOriginalFormat
vvwiOriginalFormat,
//! see VVWINFO::szPerformers
vvwiPerformers,                         // <-- 40
//! see VVWINFO::szProducer
vvwiProducer,
//! see VVWINFO::szProduct
vvwiProduct,
//! see VVWINFO::szSoftware
vvwiSoftware,
//! see VVWINFO::szSpecialPlaybackRequirements
vvwiSpecialPlaybackRequirements,
//! see VVWINFO::szTrack
vvwiTrack,
```

```
//! see VVWINFO::szWarning
vvwiWarning,
//! see VVWINFO::szURLLink
vvwiURLLink,
//! see VVWINFO::szEditData1
vvwiEditData1,
//! see VVWINFO::szEditData2
vvwiEditData2,
//! see VVWINFO::szEditData3
vvwiEditData3,                          // <-- 50
//! see VVWINFO::szEditData4
vvwiEditData4,
//! see VVWINFO::szEditData5
vvwiEditData5,
//! see VVWINFO::szEditData6
vvwiEditData6,
//! see VVWINFO::szEditData7
vvwiEditData7,
//! see VVWINFO::szEditData8
vvwiEditData8,
//! see VVWINFO::szEditData9
vvwiEditData9,
//! see VVWINFO::szVersionString
vvwiVersionString,
//! see VVWINFO::szManufacturer
vvwiManufacturer,
//! see VVWINFO::szLanguage
vvwiLanguage,
//! see VVWINFO::szFormat
vvwiFormat,                             // <-- 60
//! see VVWINFO::szInputDevice
vvwiInputDevice,
//! see VVWINFO::szDeviceModelNum
vvwiDeviceModelNum,
//! see VVWINFO::szDeviceSerialNum
vvwiDeviceSerialNum,
//! see VVWINFO::szReel
vvwiReel,
//! see VVWINFO::szShot
vvwiShot,
//! see VVWINFO::szTake
vvwiTake,
//! see VVWINFO::szSlateInfo
vvwiSlateInfo,
//! see VVWINFO::szFrameAttribute
vvwiFrameAttribute,
//! see VVWINFO::szEpisode
vvwiEpisode,
//! see VVWINFO::szScene
vvwiScene,                              // <-- 70
//! see VVWINFO::szDailyRoll
vvwiDailyRoll,
//! see VVWINFO::szCamRoll
vvwiCamRoll,
```

```
//! see VVWINFO::szSoundRoll
vvwiSoundRoll,
//! see VVWINFO::szLabRoll
vvwiLabRoll,
//! see VVWINFO::szKeyNumberPrefix
vvwiKeyNumberPrefix,
//! see VVWINFO::szInkNumberPrefix
vvwiInkNumberPrefix,
//! see VVWINFO::szPictureIcon
vvwiPictureIcon,
//! see VVWINFO::szProxyFile
vvwiProxyFile,
//!
vvwiCustomMetadataBlockPointer,
//!
vvwiImageInfo,
//!
vvwiUMID,
//
vvwiEND_OF_STRINGS,

vvwiNumericStart = 0x1000,
//! see VVWINFO::dwTimeCode
vvwiTimeCode,
//! see VVWINFO::dwUserBits
vvwiUserBits,
//! see VVWINFO::dwVITCTimeCode
vvwiVITCTimeCode,
//! see VVWINFO::dwVITCUserBits
vvwiVITCUserBits,
//! see VVWINFO::dwVITCLine3
vvwiVITCLine3,
//! see VVWINFO::dwPosterFrame
vvwiPosterFrame,
//! see VVWINFO::dwAFrame
vvwiAFrame,
//! see VVWINFO::dwAspectRatio
vvwiAspectRatio,
//! see VVWINFO::dwOriginalRate
vvwiOriginalRate,
//! see VVWINFO::dwOriginalScale
vvwiOriginalScale,
//! see VVWINFO::dwConversions
vvwiConversions,
//! see VVWINFO::dwVersionNumber
vvwiVersionNumber,
//! see VVWINFO::dwFileSize
vvwiFileSize,
//! see VVWINFO::dwFileDate
vvwiFileDate,
//! see VVWINFO::dwFileTime
vvwiFileTime,
//! see VVWINFO::dwSequenceNumber
vvwiSequenceNumber,
```

```
        //! see VVWINFO::dwTotalStreams
        vvwiTotalStreams,
        //! see VVWINFO::dwTotalLength
        vvwiTotalLength,
        //! see VVWINFO::dwFilmManufacturerCode
        vvwiFilmManufacturerCode,
        //! see VVWINFO::dwFilmTypeCode
        vvwiFilmTypeCode,
        //! see VVWINFO::dwWhitePoint
        vvwiWhitePoint,
        //! see VVWINFO::dwBlackPoint
        vvwiBlackPoint,
        //! see VVWINFO::dwBlackGain
        vvwiBlackGain,
        //! see VVWINFO::dwBreakPoint
        vvwiBreakPoint,
        //! see VVWINFO::dwGamma1000
        vvwiGamma1000,
        //! see VVWINFO::dwTagNumber
        vvwiTagNumber,
        //! see VVWINFO::dwFlags
        vvwiFlags,
        //! see VVWINFO::dwTimeCodeType
        vvwiTimeCodeType,
        //! see VVWINFO::dwLTCTimeCodeType
        vvwiLTCTimeCodeType,
        //! see VVWINFO::dwVITCTimeCodeType
        vvwiVITCTimeCodeType,
        //! see VVWINFO::dwProdDate
        vvwiProdDate,
        //End: v3.0
        //! see VVWINFO::dwUniqueID
        vvwiUniqueID,
        //!
        vvwiCustomMetadataBlockType,
        vvwiCustomMetadataBlockSize,
        vvwiNorthSouthEastWest,
        vvwiLatitude,
        vvwiLongitude,
        vvwiExposure,
        vvwiRedGain,
        vvwiBlueGain,
        vvwiWhiteBalance,

        vvwiEND_OF_DWORD_V2,
        // Add elements here
        //VVVID STRUCT
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoWidth = 0x10000,
        //! XML tag name for width
#define VVWINFOTAG_woVideoWidth                         "Width"
#define VVWINFODESC_woVideoWidth                    "Width"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoHeight,
```

```
        //! XML tag name for height
#define VVWINFOTAG_woVideoHeight                    "Height"
#define VVWINFODESC_woVideoHeight                   "Height"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoPlanes,
        //! XML tag name for planes
#define VVWINFOTAG_woVideoPlanes                    "Planes"
#define VVWINFODESC_woVideoPlanes                   "Planes"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoBitCount,
        //! XML tag name for bit count
#define VVWINFOTAG_woVideoBitCount                  "BitCount"
#define VVWINFODESC_woVideoBitCount                 "BitCount"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoCompression,
        //! XML tag name for compression (fourcc)
#define VVWINFOTAG_woVideoCompression               "Compression"
#define VVWINFODESC_woVideoCompression              "Compression"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoSizeImage,
        //! XML tag name for size of the image in unsigned chars
#define VVWINFOTAG_woVideoSizeImage                 "SizeImage"
#define VVWINFODESC_woVideoSizeImage                "SizeImage"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoXPelsPerMeter,
        //! XML tag name for X pels per meter
#define VVWINFOTAG_woVideoXPelsPerMeter             "XPelsPerMeter"
#define VVWINFODESC_woVideoXPelsPerMeter    "XPelsPerMeter"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoYPelsPerMeter,
        //! XML tag name for Y pels per meter
#define VVWINFOTAG_woVideoYPelsPerMeter             "YPelsPerMeter"
#define VVWINFODESC_woVideoYPelsPerMeter    "YPelsPerMeter"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoClrUsed,
        //! XML tag name for color elements used
#define VVWINFOTAG_woVideoClrUsed                   "ClrUsed"
#define VVWINFODESC_woVideoClrUsed                  "ClrUsed"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoClrImportant,
        //! XML tag name for color important
#define VVWINFOTAG_woVideoClrImportant              "ClrImportant"
#define VVWINFODESC_woVideoClrImportant             "ClrImportant"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoReserved,
        //! XML tag name for reserved array
#define VVWINFOTAG_woVideoReserved                  "Reserved"
#define VVWINFODESC_woVideoReserved                 "Reserved"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoFccType,
        //! XML tag name for four cc type (video/audio)
#define VVWINFOTAG_woVideoFccType                   "FccType"
#define VVWINFODESC_woVideoFccType                  "FccType"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
```

```
        vvwiVideoFccHandler,
        //! XML tag name for four cc handler
#define VVWINFOTAG_woVideoFccHandler                "FccHandler"
#define VVWINFODESC_woVideoFccHandler               "FccHandler"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoFlags,
        //! XML tag name for flags
#define VVWINFOTAG_woVideoFlags                             "Flags"
#define VVWINFODESC_woVideoFlags                    "Flags"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoCaps,
        //! XML tag name for capabilities
#define VVWINFOTAG_woVideoCaps                              "Caps"
#define VVWINFODESC_woVideoCaps                             "Caps"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoPriority,
        //! XML tag name for priority
#define VVWINFOTAG_woVideoPriority                  "Priority"
#define VVWINFODESC_woVideoPriority                 "Priority"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoLanguage,
        //! XML tag name for language
#define VVWINFOTAG_woVideoLanguage                  "Language"
#define VVWINFODESC_woVideoLanguage                         "Language"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoScale,
        //! XML tag name for scale (fps = rate / scale)
#define VVWINFOTAG_woVideoScale                             "Scale"
#define VVWINFODESC_woVideoScale                    "Scale"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoRate,
        //! XML tag name for rate (fps = rate / scale)
#define VVWINFOTAG_woVideoRate                              "Rate"
#define VVWINFODESC_woVideoRate                             "Rate"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
    vvwiVideoStart,
        //! XML tag name for start frame
#define VVWINFOTAG_woVideoStart                             "Start"
#define VVWINFODESC_woVideoStart                    "Start"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoLength,
        //! XML tag name for the length in frames
#define VVWINFOTAG_woVideoLength                    "Length"
#define VVWINFODESC_woVideoLength                   "Length"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoInitialFrames,
        //! XML tag name for number of initial frames to load
#define VVWINFOTAG_woVideoInitialFrames             "InitialFrames"
#define VVWINFODESC_woVideoInitialFrames    "InitialFrames"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
    vvwiVideoSuggestedBufferSize,
        //! XML tag name for suggested maximum buffer size
#define VVWINFOTAG_woVideoSuggestedBufferSize       "SuggestedBufferSize"
#define VVWINFODESC_woVideoSuggestedBufferSize      "SuggestedBufferSize"
```

```
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoQuality,
        //! XML tag name for quality
#define VVWINFOTAG_woVideoQuality                   "Quality"
#define VVWINFODESC_woVideoQuality                  "Quality"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoSampleSize,
        //! XML tag name for recommended sample size
#define VVWINFOTAG_woVideoSampleSize                "SampleSize"
#define VVWINFODESC_woVideoSampleSize               "SampleSize"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoEditCount,
        //! XML tag name for number of edits done on this file
#define VVWINFOTAG_woVideoEditCount                 "EditCount"
#define VVWINFODESC_woVideoEditCount                "EditCount"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoFormatChangeCount,
        //! XML tag name for number of format changes
#define VVWINFOTAG_woVideoFormatChangeCount
        "FormatChangeCount"
#define VVWINFODESC_woVideoFormatChangeCount    "FormatChangeCount"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoPitch,
        //! XML tag name for video line pitch
#define VVWINFOTAG_woVideoPitch                         "Pitch"
#define VVWINFODESC_woVideoPitch                    "Pitch"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoDrFlags,
        //! XML tag name for internal drastic flags
#define VVWINFOTAG_woVideoDrFlags                   "DrFlags"
#define VVWINFODESC_woVideoDrFlags                  "DrFlags"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoFileType,
        //! XML tag name for drastic 'mft' file type
#define VVWINFOTAG_woVideoFileType                  "FileType"
#define VVWINFODESC_woVideoFileType                 "FileType"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoResDrastic,
        //! XML tag name for reserved drastic array of DWORDs
#define VVWINFOTAG_woVideoResDrastic        "ResDrastic"
#define VVWINFODESC_woVideoResDrastic               "ResDrastic"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiAudioType,
        //! XML tag
#define VVWINFOTAG_woAudioType                          "AudioType"
#define VVWINFODESC_woAudioType                         "AudioType"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiAudioChannels,
        //! XML tag
#define VVWINFOTAG_woAudioChannels                  "AudioChannels"
#define VVWINFODESC_woAudioChannels                 "AudioChannels"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiAudioFrequency,
        //! XML tag
```

```
#define VVWINFOTAG_woAudioFrequency                 "AudioFrequency"
#define VVWINFODESC_woAudioFrequency                "AudioFrequency"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiAudioBits,
        //! XML tag
#define VVWINFOTAG_woAudioBits                                  "AudioBits"
#define VVWINFODESC_woAudioBits                                 "AudioBits"
        //char  szName[_VVWXXX_NAME_SIZE];   // Stream identifier
        //RECT/*16*/  rcFrame;                                   // Frame dimensions
        vvwiLastElementPlus1
        // DO NOT ADD ANYTHING BELOW vvwiLastElementPlus1
};
```

Direct Link Header

dtmediaread.h

```
/
**********************************************************************
*****
 *
 *
 * Copyright (c) 1998-2023 Drastic Technologies Ltd.  All Rights Reserved.
 * 523 The Queensway, Suite 201 Toronto ON M8Y 1J7
 * phone (416) 255 5636  fax (416) 255 8780
 * engineering@drastictech.com      http://www.drastic.tv

**********************************************************************
****/
// drmediaread.h : Declaration of the dtmediaread api

// Hacking class from activex control

#ifndef __DTMEDIAREAD_DRASTIC_API_9204jrewf348j4_H_
#define __DTMEDIAREAD_DRASTIC_API_9204jrewf348j4_H_

////////////////////////////////////////////////////////////////////////////

#define DTMRHANDLE void*

#ifdef _WIN32
#define DTMRCALLTYPE __stdcall
#include <windows.h>
#else
#define DTMRCALLTYPE
#include <stddef.h>
#endif

#ifdef __cplusplus
extern "C" {         // PREVENT C++ NAME-MANGLING
#endif

/** The read types
 */
//! Windows RGBA 8 bits per component, 32 total (like bitmap, tga, etc.)
const unsigned long DTMR_READTYPE_ARGB = 0;
//! RGB 8 bits per component, 24 total
const unsigned long DTMR_READTYPE_RGB = 0x10000000;
//! Alpha only 8 bits per component, repeated to 24
const unsigned long DTMR_READTYPE_AAA = 0x20000000;
//! 8 Bit YCbCr (yuv2, D1/HDSDI raw 4:2:2 video
const unsigned long DTMR_READTYPE_UYVY = 1;
//! 10 Bit v210 (quicktime packing) 4:2:2 video
const unsigned long DTMR_READTYPE_V210 = 2;
//! 10 Bit RGB 4:4;4 (dpx packing)
const unsigned long DTMR_READTYPE_RGB10Bit = 3;
//! 16 bit per component (64 bit) RGBA 4:4:4:4
```

```
const unsigned long DTMR_READTYPE_RGBA64 = 4;
//! RGB 16 bits per component, 48 total
const unsigned long DTMR_READTYPE_RGB48 = 0x10000004;
//! Alpha only 16 bits per component, repeated to 48
const unsigned long DTMR_READTYPE_AAA16 = 0x20000004;
//! 16 bit half float per component RGBA (GPU)
const unsigned long DTMR_READTYPE_RGBAHALFFLOAT = 5;
//! 16 bit half float per component RGB (GPU)
const unsigned long DTMR_READTYPE_RGBHALFFLOAT = 6;
//! Set to invert the picture vertically
const unsigned long DTMR_READFLAG_FLIP = 0x80000000;
//! Invalid file
const unsigned long DTMR_READTYPE_INVALID = -1;
//! Set readtype to video frame size AUDIO to 16 bits LE
const unsigned long DTMR_READTYPE_FRAME_AUDIO_16LE = (0x00010000 | 16);
//! Set readtype to video frame size AUDIO to 32 bits (note, 16, 20, 24 will be shifted to most significant, LE)
const unsigned long DTMR_READTYPE_FRAME_AUDIO_32LE = (0x00010000 | 32);
//! Set readtype to arbitrary sample AUDIO to 16 bits LE
const unsigned long DTMR_READTYPE_SAMPLE_AUDIO_16LE = (0x00110000 | 16);
//! Set readtype to arbitrary sample AUDIO to 32 bits (note, 16, 20, 24 will be shifted to most significant, LE)
const unsigned long DTMR_READTYPE_SAMPLE_AUDIO_32LE = (0x00110000 | 32);



/** Open a new file, stream, or network source for preview
 */
DTMRHANDLE DTMRCALLTYPE dtmrOpen(char * szFileName, unsigned long dwFlags);
typedef DTMRHANDLE (DTMRCALLTYPE * p_dtmrOpen)(char * szFileName, unsigned long dwFlags);

/** Special case:  Open a video and array of audio files
*  szFileNameVAA[0] = video file name
*  szFileNameVAA[1] = first audio file
*  szFileNameVAA[n] = last audio file
*  szFileNameVAA[n+1] = NULL for rest of audio entries
*/
DTMRHANDLE DTMRCALLTYPE dtmrOpenMulti(char * szFileNameVAA[17], unsigned long dwFlags);

/** Close the currently open stream or file
 */
long DTMRCALLTYPE dtmrClose(DTMRHANDLE dtmr);
typedef long (DTMRCALLTYPE * p_dtmrClose)(DTMRHANDLE dtmr);

/** Returns recommended and supported read types
 */
long DTMRCALLTYPE dtmrGetReadTypes(DTMRHANDLE dtmr, unsigned long dwIndex, unsigned long * pdwTypes);
typedef long (DTMRCALLTYPE * p_dtmrGetReadTypes)(DTMRHANDLE dtmr, unsigned long dwIndex, unsigned long * pdwTypes);
```

```
/** The final file name used for the source file
 */
long DTMRCALLTYPE dtmrSourceFileName(DTMRHANDLE dtmr, char *
tszMAX_PATHString);
typedef long (DTMRCALLTYPE * p_dtmrSourceFileName)(DTMRHANDLE dtmr, char *
tszMAX_PATHString);

/** Source video media's height
 */
long DTMRCALLTYPE dtmrSourceHeight(DTMRHANDLE dtmr, long *pVal);
typedef long (DTMRCALLTYPE * p_dtmrSourceHeight)(DTMRHANDLE dtmr, long
*pVal);

/** Source video media's width
 */
long DTMRCALLTYPE dtmrSourceWidth(DTMRHANDLE dtmr, long *pVal);
typedef long (DTMRCALLTYPE * p_dtmrSourceWidth)(DTMRHANDLE dtmr, long
*pVal);

/* Source video media's bit depth
 */
long DTMRCALLTYPE dtmrSourceBitDepth(DTMRHANDLE dtmr, long *pVal);
typedef long (DTMRCALLTYPE * p_dtmrSourceBitDepth)(DTMRHANDLE dtmr, long
*pVal);

/* Source video media's fourcc compression code
 */
long DTMRCALLTYPE dtmrSourceFourCC(DTMRHANDLE dtmr, long *pVal);
typedef long (DTMRCALLTYPE * p_dtmrSourceFourCC)(DTMRHANDLE dtmr, long
*pVal);

/* Source video media's bit rate
 */
long DTMRCALLTYPE dtmrSourceBitRate(DTMRHANDLE dtmr, long *pVal);
typedef long (DTMRCALLTYPE * p_dtmrSourceBitRate)(DTMRHANDLE dtmr, long
*pVal);

/* Source video media's frame size for the requested or current frame
 */
long DTMRCALLTYPE dtmrSourceFrameSize(DTMRHANDLE dtmr, long dwFrame, long
*pVal);
typedef long (DTMRCALLTYPE * p_dtmrSourceFrameSize)(DTMRHANDLE dtmr, long
dwFrame, long *pVal);

/* Source video total channels
 */
long DTMRCALLTYPE dtmrSourceVideoChannels(DTMRHANDLE dtmr, long *pVal);
typedef long (DTMRCALLTYPE * p_dtmrSourceVideoChannels)(DTMRHANDLE dtmr,
long *pVal);

/* Source audio total channels
 */
long DTMRCALLTYPE dtmrSourceAudioChannels(DTMRHANDLE dtmr, long *pVal);
typedef long (DTMRCALLTYPE * p_dtmrSourceAudioChannels)(DTMRHANDLE dtmr,
```

```
long *pVal);

/** Source audio media frequency
 */
long DTMRCALLTYPE dtmrSourceAudioFrequency(DTMRHANDLE dtmr, long *pVal);
typedef long (DTMRCALLTYPE * p_dtmrSourceAudioFrequency)(DTMRHANDLE dtmr,
long *pVal);

/** Source audio media bits per sample
 */
long DTMRCALLTYPE dtmrSourceAudioBitsPerSample(DTMRHANDLE dtmr, long
*pVal);
typedef long (DTMRCALLTYPE * p_dtmrSourceAudioBitsPerSample)(DTMRHANDLE
dtmr, long *pVal);

/* Source audio media's fourcc compression code
 */
long DTMRCALLTYPE dtmrSourceAudioFourCC(DTMRHANDLE dtmr, long *pVal);
typedef long (DTMRCALLTYPE * p_dtmrSourceAudioFourCC)(DTMRHANDLE dtmr,
long *pVal);

/** Return the duration (total number of frames) of the media
 */
long DTMRCALLTYPE dtmrDuration(DTMRHANDLE dtmr, long *pVal);
typedef long (DTMRCALLTYPE * p_dtmrDuration)(DTMRHANDLE dtmr, long *pVal);

/** Return the audio duration (total number of audio samples) of the media
 */
long DTMRCALLTYPE dtmrAudioDuration(DTMRHANDLE dtmr, long *pVal);
typedef long (DTMRCALLTYPE * p_dtmrAudioDuration)(DTMRHANDLE dtmr, long
*pVal);

/** Source video rate value (FPS = SourceRate / SourceScale)
 */
long DTMRCALLTYPE dtmrSourceRate(DTMRHANDLE dtmr, long *pVal);
typedef long (DTMRCALLTYPE * p_dtmrSourceRate)(DTMRHANDLE dtmr, long
*pVal);

/** Source video scale value (FPS = SourceRate / SourceScale)
 */
long DTMRCALLTYPE dtmrSourceScale(DTMRHANDLE dtmr, long *pVal);
typedef long (DTMRCALLTYPE * p_dtmrSourceScale)(DTMRHANDLE dtmr, long
*pVal);

/** Return source metadata information that are numeric (DWORDs or longs)
 */
long DTMRCALLTYPE dtmrSourceMetaDataDWORD(DTMRHANDLE dtmr, long
dwMetaDataElement, long *pVal);
typedef long (DTMRCALLTYPE * p_dtmrSourceMetaDataDWORD)(DTMRHANDLE dtmr,
long dwMetaDataElement, long *pVal);

/** Return source metadata information that are string data
 */
long DTMRCALLTYPE dtmrSourceMetaDataSTR(DTMRHANDLE dtmr, long
```

```
dwMetaDataElement, char * szMAX_PATHString);
typedef long (DTMRCALLTYPE * p_dtmrSourceMetaDataSTR)(DTMRHANDLE dtmr,
long dwMetaDataElement, char * szMAX_PATHString);

/** Set the read type for the video frames
 */
long DTMRCALLTYPE dtmrSetReadType(DTMRHANDLE dtmr, long lReadType);
typedef long (DTMRCALLTYPE * p_dtmrSetReadType)(DTMRHANDLE dtmr, long
lReadType);

/** Get the current absolute (zero based) Frame
 */
long DTMRCALLTYPE dtmrGetFrame(DTMRHANDLE dtmr, long *pVal);
typedef long (DTMRCALLTYPE * p_dtmrGetFrame)(DTMRHANDLE dtmr, long *pVal);

/** Set the current absolute (zero based) Frame
 */
long DTMRCALLTYPE dtmrSetFrame(DTMRHANDLE dtmr, long newVal);
typedef long (DTMRCALLTYPE * p_dtmrSetFrame)(DTMRHANDLE dtmr, long newVal);

/** Set the channel for the video frames (0, 1, 2, 3, 4 etc.) (0 = 0x03, 1 = 0x0C, 2
= 0x30, 3 = 0xC0 etc.)
 */
long DTMRCALLTYPE dtmrSetVideoChannel(DTMRHANDLE dtmr, long lVideoChannel);
typedef long (DTMRCALLTYPE * p_dtmrSetVideoChannel)(DTMRHANDLE dtmr, long
lVideoChannel);

/** Set the audio channel pair to monitor (0  = 1+2, 1 = 3+4, 2 = 5+6, 3 = 7+8
etc.)
 */
long DTMRCALLTYPE dtmrSetAudioChannelPair(DTMRHANDLE dtmr, long
lAudioChannelPair);
typedef long (DTMRCALLTYPE * p_dtmrSetAudioChannelPair)(DTMRHANDLE dtmr,
long lAudioChannelPair);

/** Return the last GetVideoFrame VITC (vertical blank) time code
 */
long DTMRCALLTYPE dtmrLastVitcFrame(DTMRHANDLE dtmr, long *pVal);
typedef long (DTMRCALLTYPE * p_dtmrLastVitcFrame)(DTMRHANDLE dtmr, long
*pVal);

/** Return the last GetVideoFrame VITC (vertical blank time code) user bits
 */
long DTMRCALLTYPE dtmrLastVitcUb(DTMRHANDLE dtmr, long *pVal);
typedef long (DTMRCALLTYPE * p_dtmrLastVitcUb)(DTMRHANDLE dtmr, long *pVal);

/** Return the last GetVideoFrame LTC (SMPTE) time code
 */
long DTMRCALLTYPE dtmrLastLtcFrame(DTMRHANDLE dtmr, long *pVal);
typedef long (DTMRCALLTYPE * p_dtmrLastLtcFrame)(DTMRHANDLE dtmr, long
*pVal);

/** Return the last GetVideoFrame LTC (SMPTE time code) user bits
 */
```

```
long DTMRCALLTYPE dtmrLastLtcUb(DTMRHANDLE dtmr, long *pVal);
typedef long (DTMRCALLTYPE * p_dtmrLastLtcUb)(DTMRHANDLE dtmr, long *pVal);

/** GetVideoFrame returns a safe array containing one video frame
 */
long DTMRCALLTYPE dtmrGetVideoFrame(DTMRHANDLE dtmr, unsigned char *
psvFrame, long * plSize);
typedef long (DTMRCALLTYPE * p_dtmrGetVideoFrame)(DTMRHANDLE dtmr,
unsigned char * psvFrame, long * plSize);

/** GetAudioFrame returns a safe array containing one video frame worth of audio
data
 */
long DTMRCALLTYPE dtmrGetAudioFrame(DTMRHANDLE dtmr, unsigned char *
psaFrame, long * plSize);
typedef long (DTMRCALLTYPE * p_dtmrGetAudioFrame)(DTMRHANDLE dtmr,
unsigned char * psaFrame, long * plSize);

/** Get current extended data
 */
long DTMRCALLTYPE dtmrGetCurExtendedData(DTMRHANDLE dtmr, unsigned char
*pvData, unsigned long * plFlags, long *plSize);
typedef long (DTMRCALLTYPE * p_dtmrGetCurExtendedData)(DTMRHANDLE dtmr,
unsigned char *pvData, unsigned long * plFlags, long *plSize);

//! Data is EIA-608B SD closed caption data field one (uses 2 bytes)
#define FRAMEINFO_DATA_F1_EIA608              0x00000001
//! Data is EIA-608B SD closed caption data field two (uses 2 bytes)
#define FRAMEINFO_DATA_F2_EIA608              0x00000002
//! Data is EIA-708 HD closed caption data (uses remaining bytes = minus the
above)
#define FRAMEINFO_DATA_EIA708                         0x00000100

/** Get current closed captions (from last video frame loaded) including size and
flags
 */
long DTMRCALLTYPE dtmrGetCurClosedCaptions(DTMRHANDLE dtmr, unsigned char
*pvCC, long *plCCSize, long * plCCFlags);
typedef long (DTMRCALLTYPE * p_dtmrGetCurClosedCaptions)(DTMRHANDLE dtmr,
unsigned char *pvCC, long *plCCSize, long * plCCFlags);

/** Advanced - send/return a mediacmd structure
*/
long DTMRCALLTYPE dtmrSetMode(DTMRHANDLE dtmr, void * pMediCmd);
typedef long (DTMRCALLTYPE * p_dtmrSetMode)(DTMRHANDLE dtmr, void *
pMediCmd);

// dwFlags
        //! Send this in if you just need the filename (faster than getting all the info)
#define DPOSSIZENAME_FILENAME_ONLY                    0x40000000          //
Same as DFRAME_SKIP_FRAME
        //! Flag for mediafile/avhal to get audio dframe
#define GetAudio      0x00000000
        //! Flag for mediafile/avhal to get video dframe
```

```
#define GetVideo      0x00000001

// dwFrameFlags
#define DPOSSIZENAME_VIDEO_FRAME            0x00000001
        //! Is this file type currently recording
#define DPOSSIZENAME_RECORDING                      0x00000004
        //! This frame needs to be made black (default frame) in MediaFile
#define DPOSSIZENAME_PLEASE_BLACK           _PDFRAMEFLAGS_PLEASE_BLACK
        //      0x00000080
        //! This is a mono audio chunk
#define DPOSSIZENAME_MONO_AUDIO_FRAME   0x00000100
        //! This is a stereo audio chunk
#define DPOSSIZENAME_STEREO_AUDIO_FRAME         0x00000200
#define DPOSSIZENAME_QUAD_AUDIO_FRAME   0x00000400
#define DPOSSIZENAME_4_1_AUDIO_FRAME      0x00000800
#define DPOSSIZENAME_5_1_AUDIO_FRAME      0x00001000
#define DPOSSIZENAME_7_1_AUDIO_FRAME      0x00002000
#define DPOSSIZENAME_9_1_AUDIO_FRAME      0x00004000
#define DPOSSIZENAME_AUDIO_MASK
(DPOSSIZENAME_MONO_AUDIO_FRAME|DPOSSIZENAME_STEREO_AUDIO_FRAME|
DPOSSIZENAME_STEREO_AUDIO_FRAME|DPOSSIZENAME_QUAD_AUDIO_FRAME|
DPOSSIZENAME_4_1_AUDIO_FRAME|DPOSSIZENAME_5_1_AUDIO_FRAME|
DPOSSIZENAME_7_1_AUDIO_FRAME|DPOSSIZENAME_9_1_AUDIO_FRAME)
#define DPOSSIZENAME_FRAME_MASK                     0x0000FFFF
        //! This frame contains audio data see DFRAME::dwType
#define DFRAME_TYPE_AUDIO               0x00010000
        //! 16 bit audio
#define DPOSSIZENAME_AUD_16_16_BIT                  0x00100000
        //! 20 bit audio in 24
#define DPOSSIZENAME_AUD_20_24_BIT                  0x00200000
        //! 24 bit audio in 24
#define DPOSSIZENAME_AUD_24_24_BIT                  0x00400000
        //! 24/32 bit audio in 32
#define DPOSSIZENAME_AUD_24_32_BIT                  0x00800000
        //! 32/32 bit audio in 32
#define DPOSSIZENAME_AUD_32_32_BIT                  0x01000000
        //! Audio is compressed
#define DPOSSIZENAME_AUD_COMPRESSED                 0x02000000
        //! Audio is big endian, else little endian
#define DPOSSIZENAME_AUD_BIGENDIAN_BIT   0x00080000
        //! Just for completeness
#define DPOSSIZENAME_AUD_LITTLEENDIAN_BIT       0x00000000
        //! This frame is independent of other frames for decode see
DFRAME::dwType
#define DFRAME_TYPE_KEYFRAME  0x10000000
        //! This frame is independent of other frames for decode (an MPEG I Frame)
see DFRAME::dwType
#define DFRAME_TYPE_KEYFRAME_I       0x10000000
        //! This frame requires previous keyframe(s) (for MPEG a P Frame) see
DFRAME::dwType
#define DFRAME_TYPE_KEYFRAME_P       0x80000000
        //! This frame requires more than one frame to decode (for MPEG a B Frame)
see DFRAME::dwType
#define DFRAME_TYPE_KEYFRAME_B       0x20000000
```

```
        //! This frame should be skipped (decoded, but not displayed) - Used to reach
seek frame on a non key frame from key frame see DFRAME::dwType
#define DFRAME_SKIP_FRAME                0x40000000

/** Get info on a frame of audio or video
*/
long DTMRCALLTYPE dtmrGetFileFrameInfo(DTMRHANDLE dtmr, unsigned long
dwFrame, unsigned long dwChannels, unsigned long dwFlags,
                                                                    size_t *
pnPosition, size_t * pnSize, unsigned long * pdwFrameFlags,
                                                                    char *
szFilePathAndName);
typedef long (DTMRCALLTYPE * p_dtmrGetFileFrameInfo)(DTMRHANDLE dtmr,
unsigned long dwFrame, unsigned long dwChannels, unsigned long dwFlags,
                                                                    size_t *
pnPosition, size_t * pnSize, unsigned long * pdwFrameFlags,
                                                                    char *
szFilePathAndName);

#ifdef __cplusplus
}                       // PREVENT C++ NAME-MANGLING
#endif

//////////////////////////////////////////////////////////////////////////

#endif //__DTMEDIAREAD_DRASTIC_API_9204jrewf348j4_H_
```

This manual has been compiled to assist the user in their experience using the **Drastic DTMediaRead SDK**. It is believed to be correct at the time of writing, and every effort has been made to provide accurate and useful information. Any errors that may have crept in are unintentional and will hopefully be purged in a future revision of this document. We welcome your feedback.

Drastic Technologies Ltd
523 The Queensway, Suite 201
Toronto, ON, M8Y 1J7
Canada
(416) 255 5636
(416) 255 8780