TWAIN Errata

For Version 2.0

March 11th, 2009

Purpose

The Errata Document identifies omissions or mistakes in the TWAIN Specification. This information may change before being ratified into a future version of the TWAIN Specification.

Items marked [TBD] need clarification about content or positioning in the Specification.

History

Date	Comment
August 20th, 2008	Initial version
August 25th, 2008	Added CAP_MICRENABLED
January 20th, 2009	Added DAT_CAPABILITY clarification and TWCC_CHECKSTATUS
February 2nd, 2009	Addressed all TBD items.
March 3rd, 2009	Checked items against 2.1 Spec and indicated corrections.
March 9th, 2009	Checked items against 2.1 Spec and indicated corrections.
March 10th, 2009	Checked items against 2.1 Spec and indicated corrections.
March 11th, 2009	Checked items against 2.1 Spec, document complete.

Contents

DAT_ENTRYPOINT [DONE]

This text replaces that under the heading "Identifying TWAIN 2.0 Elements" in Chapter 2.

Identifying TWAIN 2.0 Elements

It is not sufficient to test the TW_IDENTITY.ProtocolMajor field to determine if an Application, a Data Source Manager or a Source is TWAIN 2.0 compliant. Check the TW_IDENTITY.SupportedGroups field for the Application or the Source, and look for the following:

- DF_APP2, indicating that the Application is 2.0 compliant
- DF_DSM2, indicating that the Data Source Manager is 2.0 compliant
- DF_DS2, indicating that the Data Source is 2.0 compliant

Applications

All TWAIN 2.0 compliant Applications must report DF_APP2 in their TW_IDENTITY.SupportedGroups field.

All TWAIN 2.0 compliant Applications must test for the DF_DSM2 flag in the TW_IDENTITY.SupportedGroups field, after a call to DG_CONTROL / DAT_PARENT / MSG_OPENDSM. If this flag is not found, then follow the legacy behavior for 1.x Applications, using the memory management functions detailed in the TWAIN Specification.

If the flag is found, then the Application must call DG_CONTROL / DAT_ENTRYPOINT / MSG_GET in State 3, before performing any other operation, to obtain pointers to the memory management functions.

Sources

All TWAIN 2.0 compliant Sources must report DF_DS2 in their TW_IDENTITY.SupportedGroups field.

All TWAIN 2.0 compliant Sources must be prepared to receive the DG_CONTROL / DAT_ENTRYPOINT / MSG_SET call in State 3, before DG_CONTROL / DAT_IDENTITY / MSG_OPENDS is called. If this operation is not called, then follow the legacy behavior for 1.x Sources, using the memory management functions detailed in the TWAIN Specification, and locating the Data Source Manager as indicated.

If the operation is called then the Source must use the pointers to the memory management functions, and must use the supplied entry point to access DSM_Entry.

DAT_CALLBACK [DONE]

Place this after the "Identifying TWAIN 2.0 Elements" in Chapter 2..

Using DAT_CALLBACK to Messages from the Source to the Application

Applications

TWAIN Applications running on Linux or Apple Macintosh OS X must use DG_CONTROL / DAT_CALLBACK / MSG_REGISTERCALLBACK to register to receive asynchronous notifications for events MSG_XFERREADY.

TWAIN Applications on Microsoft Windows that detect the presence of the DF_DSM2 flag inside of TW_IDENTITY.SupportedGroups are encouraged to use DAT_CALLBACK instead of processing TWRC_DSEVENT from DG_CONTROL / DAT_EVENT / MSG_PROCESSEVENT. The legacy TWAIN 1.x behavior is still supported by the Data Source Manager.

TWAIN Applications on Microsoft Windows using older versions of the Data Source Manager (no DF_DSM2 flag detected) must use the legacy behavior.

Please note that TWAIN Applications are advised to return as soon as possible from a callback function. Events like MSG_XFERREADY should initiate the image transfer in a different thread so that the callback can return immediately.

Sources

TWAIN sources running on Apple Macintosh OS X that use a Data Source Manager that does not report DF_DSM2 in TW_IDENTITY.SupportedGroups must use DG_CONTROL / DAT_CALLBACK / MSG_INVOKECALLBACK to return events like MSG_XFERREADY.

All other TWAIN sources, regardless of version or operating system must use DG_CONTROL / DAT_NULL with the appropriate message to return events like MSG_XFERREADY.

Operating System Support [DONE]

Place this after the item in the previous page, just before "Memory Management in TWAIN 2.0 and Higher"

Installation of the Data Source Manager

Microsoft Windows

Please refer to the TWAIN 1.9 Specification for support of versions of Microsoft Windows prior to Windows 2000.

Microsoft provides TWAIN_32.DLL for access to the legacy 1.x Data Source Manager on 32-bit systems and SYSWOW64.

TWAIN_32.DLL is not available for native 64-bit Applications. A native 64-bit TWAIN session must use TWAINDSM.DLL.

Applications that wish to use TWAINDSM.DLL, for access to the new open source Data Source Manager, must install it themselves. Please refer to the TWAIN website http://www.twain.org to obtain this file, and for installation instructions. This DSM is fully backwards compatible with all versions of TWAIN.

Apple Macintosh

Please refer to the TWAIN 1.9 Specification for support of operating systems prior to Apple Mac OS X.

Apple provides /System/Library/Frameworks/TWAIN.framework for access to the legacy 1.x Data Source Manager.

The TWAIN 2.0 open source Data Source Manager is not currently available on any version of Apple Mac OS X. TWAIN 2.0 compliant Application and Source writers are strongly encouraged to include and check for the DF_APP2, DF_DSM2 and DF_DS2 flags, and to handle DAT_CALLBACK when these flags are detected, so they are ready when the TWAIN 2.0 Data Source Manager appears.

Linux

Please check the TWAIN website http://www.twain.org to see if your distro is represented, and if not, please consider making a submission to the TWAIN Working Group.

Note that TWAIN 2.0 compliant Sources must support TW_USERINTERFACE.ShowUI being set to 0 (no UI), which in concert with CAP_INDICATORS set to FALSE is expected to prevent the Source from showing any dialog, making it suitable for use from any command shell.

DAT_IMAGEMEMFILEXFER [DONE]

The following information is missing from the TWAIN 2.0 Specification.

Chapter 3, after the entry for DG_IMAGE / DAT_IMAGEMEMXFER

DG_IMAGE / DAT_IMAGEMEMFILEXFER

MSG_GET	Initiate image acquisition using the Buffered Memory transfer mode, but transferring the same data one would save to a file

Chapter 4, after the section "Buffered Memory Mode Transfer"

Buffered Memory Mode Transfer With File Format

This operation works very much like Buffered Memory Mode, but the data transferred from the Source to the Application conforms to the image file format specified by a previous call to DG_IMAGE / DAT_SETUPFILEXFER / MSG_GET. There is no requirement for the data to be transferred as complete image lines or for any kind of padding, the data is assumed to be self-contained and self-describing.

Chapter 6, after the line "DAT_IMAGEMEMXFER DG_IMAGE TW_IMAGEMEMXFER structure"

DAT_IMAGEMEMFILEXFER	DG_IMAGE	TW_IMAGEMEMXFER structure

Chapter 7, after the line "DG_IMAGE DAT_IMAGEMEMXFER MSG_GET 7-126"

DG_IMAGE	DAT_IMAGEMEMFILEXFER	MSG_GET	7-???

Internationalization [DONE]

The following information overrides content in the TWAIN 2.0 Specification.

Appendix A in the TWAIN 2.0 Specification has a section on Internationalization. This information is too specific to Microsoft Windows, and is too complex to implement. It is completely deprecated and replaced by the following.

Internationalization

TWAIN 2.0 Applications and Sources use UTF-8 to pass localized data back and forth in any field with a type of TW_STR32, TW_STR64, TW_STR128 or TW_STR255. Since the first 128 characters map to US-ASCII, this requires no change for most Applications and Sources.

Some structures should never localize their data, like TW_IDENTITY, since changing locales presents a problem for Applications and Sources that need to recognize a particular string to help with customization.

The use of UTF-8 is not mandated for localized strings displayed by the Application or the Source. Those strings are internal to their respective functions, and do not pass through the TWAIN API, so the Applications and Sources are free to encode them in any way they choose.

Data Flags and Data Groups [DONE]

Place the following content in Chapter six after "Programming Basics"

Data Flags and Data Groups

Versions of the TWAIN Specification up to and including TWAIN 2.0 indicate that the high 8-bits (24 – 31) in the TW_IDENTITY.SupportedGroups are reserved for custom use.

TWAIN 2.0 has taken these bits for use by the Data Flags (DF_APP2, DF_DSM2 and DF_DS2). This breaks backwards capability with previous versions of the Specification. The risk is considered to be very low, since very few Sources or Applications work with these bits. However, the conflict can be managed in the following ways.

- Avoid the use of 0x10000000, 0x20000000 and 0x40000000, these correspond to DF_DSM2, DF_APP2 and DF_DS2. The remaining bits: 0x01000000, 0x02000000, 0x04000000, 0x08000000 and 0x80000000 are still in the custom space for Applications and Sources, and they will remain free for that use in all subsequent versions of TWAIN.
- Applications can modify their code to recognize when these bits are in use by a particular Source, which has always been a necessary pre-requisite for custom features, since the bits are guaranteed to have different meaning for different vendors.
- These flags are of most interest to the Data Source Manager, which is now open source (they dictate when DAT_ENTRYPOINT is called). If a legacy driver is using one of the custom bits, then propose a possible work-around to the TWAIN Working Group.

Condition Codes [DONE]

There is an extra TWCC_PAPERJAM line that can be removed…

Please sort the list of condition codes…

The following Chapter 10 items need to be updated.

Under "An Overview of Return Codes and Condition Codes"

The following operations can only return TWRC_SUCCESS or TWRC_FAILURE / TWCC_SEQERROR, if called in the wrong state. This is to avoid a situation where an Application is unable to shutdown a Source because of an error state, like the device being offline. The Source must comply with the request to change states.

DG_CONTROL / DAT_PENDINGXFERS / MSG_ENDXFER
DG_CONTROL / DAT_PENDINGXFERS / MSG_RESET
DG_CONTROL / DAT_USERINTERFACE / MSG_DISABLEDS
DG_CONTROL / DAT_IDENTITY / MSG_CLOSEDS
DG_CONTROL / DAT_IDENTITY / MSG_CLOSEDSM

When an Application receives this condition code, it alerts the user (so they can exit, if they wish). While waiting for the user response the Application polls the value of CAP_DEVICEONLINE. The device continues to be offline as long as this call returns TWCC_SUCCESS, with a value of FALSE.

The state 3 operation DG_CONTROL / DAT_IDENTITY / MSG_OPENDS is the only one capable of returning TWCC_CHECKDEVICEONLINE. The Application cannot check CAP_DEVICEONLINE (since that is a state 4 operation), however, it can retry the MSG_OPENDS call, if it chooses.

Under "Currently Defined Condition Codes" add this item

TWCC_CHECKDEVICEONLINE	Check the device status using CAP_DEVICEONLINE, this condition code can be returned by any TWAIN operation in state 4 or higher, or from the state 3 DG_CONTROL / DAT_IDENTITY / MSG_OPENDS. The state remains unchanged. If in state 4 the Application can poll with CAP_DEVICELINE until the value returns TRUE.

Modify this existing item.

TWCC_PAPERJAM	Transfer failed because of a feeder error, this can be returned by any of the DAT_IMAGE*XFER operations. When received the current TWAIN state remains unchanged.

This item is currently mislabeled as "TWCC_DOUBLEFEED" and it needs modifying

TWCC_PAPERDOUBLEFEED	Transfer failed because of a feeder error, this can be returned by any of the DAT_IMAGE*XFER operations. When received the current TWAIN state remains unchanged.

CAP_MICRENABLED [DONE]

In Chapter 9, under the entry for this capability, replace the entire description with the following.

Description

Get this capability to determine if the Source supports check scanning. If set to TRUE check scanning is enabled, if set to FALSE check scanning is disabled.

DAT_CAPABILITY Container Clarification [DONE]

Place this in Chapter 5, appending it to the end of the section labeled "Responding to Requests to Set Capabilities"

A Source support MSG_SET operations using the same containers it returns through MSG_GET, MSG_GETCURRENT and MSG_GETDEFAULT operations.

Example #1, a call to DG_CONTROL / DAT_CAPABILITY / MSG_GET returns a TW_ENUMERTION container. The application changes the CurrentIndex and uses DG_CONTROL / DAT_CAPABILITY / MSG_SET to update the capability.

Example #2, a call to DG_CONTROL / DAT_CAPABILITY / MSG_GET returns a TW_RANGE container. The application changes the CurrentValue and uses DG_CONTROL / DAT_CAPABILITY / MSG_SET to update the capability.

This does not imply or require support for constraining capabilities, the Source is only obligated to update the current value of the capability. If the Source does not support constraints for a capability, and the constraining values have been changed by the application, then the Source should apply the current value according to its own constraints, and if that value is valid, return TWRC_CHECKSTATUS to alert that application that it needs to do a MSG_GET to validate its changes.

Example #3, if a Source supports the following range for ICAP_BRIGHTNESS: -1000.0 to -1000.0 in steps of 20.0, and if the current value is 0.0, then a call to DG_CONTROL / DAT_CAPABILITY / MSG_SET results in the following:

twrange.ItemType = TWTY_FIX32
twrange.MinValue = -1000.0
twrange.MaxValue = 1000.0
twrange.StepSize = 20.0
twrange.DefaultValue = 0.0
twrange.CurrentValue = 0.0

If the application sets twrange.CurrentValue to 900.0 and sends this structure to the Source using DG_CONTROL / DAT_CAPABILITY / MSG_SET, the call succeeds and returns TWRC_SUCCESS.

If the application sets both twrange.CurrentValue and twrange.MaxValue to 900.0, then the status return depends on the Source. A Source that supports constaints accepts the new value and limits MaxValue to 900.0. A Source that does not support contraints accepts the value 900.0, because it falls in the range of -1000 to 1000, step 20; but it returns TWRC_CHECKSTATUS because it was unable to accept the request to limit MaxValue to 900.0.

TWCC_CHECKSTATUS [DONE]

Change all occurrences of TWCC_CHECKSTATUS to TWRC_CHECKSTATUS, TWRC_FAILURE should be removed, if it's present.

Remove all references to DAT_SETUPFILEXFER2 and TW_SETUPFILEXFER2 [DONE]

Replace "DG_CONTROL / DAT_SETUPFILEXFER2 operations" with "operation" near the bottom of page 4-18

Remove this entire block from Chapter 3.

DG_CONTROL / DAT_SETUPFILEXFER2

MSG_GET Return info about the file that the Source will write the acquired data into
MSG_GETDEFAULT Return the default file transfer information
MSG_RESET Reset current file information to default values
MSG_SET Set file transfer information for next file transfer

Remove "or DG_CONTROL / DAT_SETUPFILEXFER2" in Chapter 4

To set up the transfer the DG_CONTROL / DAT_SETUPFILEXFER or DG_CONTROL / DAT_SETUPFILEXFER2 operations of MSG_GET, MSG_GETDEFAULT, and MSG_SET can be used.

Remove this entire block in Chapter 4.

Macintosh developers must use the TW_SETUPFILEXFER2 structure (along with DAT_SETUPFILEXFER2) for TWAIN versions 1.9 and higher:

```
typedef struct {
        TW_MEMREF FileName; /* File to contain data */
        TW_UINT16 FileNameType; /* TWTY_STR1024 or TWTY_UNI512 */
        TW_UINT16 Format; /* A TWFF_xxxx constant */
        TW_HANDLE VrefNum; /* Used for Macintosh only */
        TW_UINT32 parID; /* Used for Macintosh only */

} TW_SETUPFILEXFER2, FAR *pTW_SETUPFILEXFER2;
```

Remove "or TW_SETUPFILEXFER2" in Chapter 4.

1. Allocate the required TW_SETUPFILEXFER or TW_SETUPFILEXFER2 structure. Then fill in the appropriate fields:

Remove from item 'a' in Chapter 4.

If using the TW_SETUPFILEXFER2 structure, be sure to allocate the space needed for a TWTY_STR1024 or a TWTY_UNI512 first

Remove all of item 'b'; remove all of item 'e' in Chapter 4.

Remove "or the DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_SET" in Chapter 4.

2. Invoke the DG_CONTROL / DAT_SETUPFILEXFER / MSG_SET or the DG_CONTROL /

DAT_SETUPFILEXFER2 / MSG_SET, as appropriate.

Remove the duplicate "or the DG_CONTROL / DAT_SETUPFILEXFER / MSG_GET" in Chapter 4.
After the application receives the MSG_XFERREADY notice from the Source and has issued the DG_CONTROL / DAT_SETUPFILEXFER / MSG_GET or the DG_CONTROL / DAT_SETUPFILEXFER / MSG_GET operation.:

Remove "or the DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_GET" in Chapter 5.
The Source selects a default file format and file name (typically, TWAIN.TMP in the current directory). It reports this information to the application in response to the DG_CONTROL / DAT_SETUPFILEXFER / MSG_GET or the DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_GET operation.

Remove "the DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_GET" in Chapter 5.
The application may determine all of the Source's supported file formats by using the ICAP_IMAGEFILEFORMAT capability. Based on this information, the application can request a particular file format and define its own choice of file name for the transfer. The desired file format and file name will be communicated to the Source in a DG_CONTROL / DAT_SETUPFILEXFER / MSG_GET or the DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_GET operation.

Remove the following from Chapter 6.
DAT_SETUPFILEXFER2 DG_CONTROL TW_SETUPFILEXFER2 structure

Remove "or the DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_SET" in Chapter 7.
No special set up or action required. Application should have already invoked the DG_CONTROL / DAT_SETUPFILEXFER / MSG_SET or the DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_SET operation, unless the Source's default transfer format and file name (typically, TWAINAUD.TMP) are acceptable to the application. The application need only invoke this operation once per image transferred.

Remove "the DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_SET" in Chapter 7.
Source should acquire the audio data, format it, create any appropriate header information, and
write everything into the file specified by the previous DG_CONTROL / DAT_SETUPFILEXFER / MSG_SET or the DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_SET operation, and close the file.

Remove entire section in Chapter 7 under the heading…
DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_GET

Remove entire section in Chapter 7 under the heading…
DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_GETDEFAULT

Remove entire section in Chapter 7 under the heading…
DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_RESET

Remove entire section in Chapter 7 under the heading…
DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_SET

Remove "or the DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_SET" in Chapter 7.
This operation acts on NULL data. File information can be set with the DG_CONTROL /

DAT_SETUPFILEXFER / MSG_SET or the DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_SET operation.

Remove "or the DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_SET" in Chapter 7.
No special set up or action required. Application should have already invoked the DG_CONTROL / DAT_SETUPFILEXFER / MSG_SET or the DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_SET operation unless the Source's default transfer format and file name (typically, TWAIN.TMP) are acceptable to the application. The application need only invoke this operation once per image transferred.

Remove "or the DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_SET" in Chapter 7.
Note: Applications can specify a unique file for each transfer using DG_CONTROL / DAT_SETUPFILEXFER / MSG_SET or the DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_SET operation in State 6 or 5 (and 4, of course).

Remove "or the DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_SET" in Chapter 7.
Acquire the image data, format it, create any appropriate header information, and write everything into the file specified by the previous DG_CONTROL / DAT_SETUPFILEXFER / MSG_SET or the DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_SET operation, and close the file.

Remove "or the DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_SET" in Chapter 10.
Be sure to use the DG_CONTROL / DAT_SETUPFILEXFER / MSG_SET or the DG_CONTROL / DAT_SETUPFILEXFER2 / MSG_SET operation to specify the format to be used for a particular acquisition.

Remove completely from Appendix A
DAT_SETUPFILEXFER2 selects the audio file format

Remove "or DAT_SETUPFILEXFER2" from Appendix A.
ICAP_XFERMECH selects the way an image is transferred from the Source to an Application, which has an impact on some of the characteristics of an image, which is why this value must be selected first. If TWSX_NATIVE is selected, then no other action related to image transfer is needed. If TWSX_FILE or TWSX_FILE2 is selected, then the application should negotiate ICAP_IMAGEFILEFORMAT, which will be used when DAT_SETUPFILEXFER or DAT_SETUPFILEXFER2 is called. If TWSX_MEMORY is selected, then DAT_SETUPMEMXFER will need to be called. The Application may then opt to negotiate ICAP_TILES.

Remove all references to TW_SETUPAUDIOFILEFORMAT [DONE]

Remove all references to DAT_SETUPAUDIOXFER [DONE]

Remove all references to ACAP_AUDIOFILEFORMAT [DONE]

Remove all occurrences of "ACAP_AUDIOFILEFORMAT" from Chapter 7, an example is below.
Capabilities -ICAP_XFERMECH, ICAP_IMAGEFILEFORMAT,
ACAP_XFERMECH, ACAP_AUDIOFILEFORMAT

Remove this line in Chapter 10.
ACAP_AUDIOFILEFORMAT Informs application which audio file formats the source can generate.

Remove the section under this heading in Chapter 10.
ACAP_AUDIOFILEFORMAT

Remove the following line in Appendix A.
ACAP_AUDIOFILEFORMAT negotiate available audio file formats

Remove the sentence "If TWSX_FILE or TWSX_FILE2 is selected, then ACAP_AUDIOFILEFORMAT should be negotiated."
The availability of the audio capabilities can be inferred from the presence of DG_AUDIO. If it is
available then the Application should negotiate ACAP_XFERMECH. If TWSX_FILE or TWSX_FILE2 is selected, then ACAP_AUDIOFILEFORMAT should be negotiated. Note that these operations occur independently of the current value of DAT_XFERGROUP. The actual selection of an audio file format takes place in State 6 using DAT_SETUPFILEXFER, and must be preceded by a call to DAT_XFERGROUP / MSG_SET to DG_AUDIO to change the Source over to the audio data group. Sources that transfer audio data need to set the Source back to DG_IMAGE when they are done with the audio data, and ready to get image data, or exit back to State 4.

Remove this line in Appendix A.
ACAP_AUDIOFILEFORMAT n/a No default (current value is meaningless)

Last Minute Stuff [DONE]

The first couple of items are changes that require review.

Page 7-102 DG_CONTROL / DAT_STATUS / MSG_GET add to Section Source. Fills pSourceStatus->ConditionCode with its current Condition Code and pSourceStatus->data with scanner specific data if datasource supports DAT_STATUSUTF8 so it can look up the message later.

.

Page 8-59 TW_STATUS add scanner specific to the description of Data.

Data Valid for TWAIN 2.1 and later. This field contains additional scanner specific data. If there is no data, then this value must be zero.

To be consistent these caps are only for images and should start with ICAP. Need to update both Proposed Changes and Spec if doing this.

CAP_AUTOMATICCROPUSESFRAME should be ICAP_AUTOMATICCROPUSESFRAME

CAP_AUTOMATICLENGTHDETECTION should be ICAP_AUTOMATICLENGTHDETECTION

[DROPPED, one this capability is in the same group as CAP_FEEDERENABLED]
CAP_AUTOMATICSENSEMEDIUM should be ICAP_ AUTOMATICSENSEMEDIUM

CAP_IMAGEMERGE should be ICAP_IMAGEMERGE

CAP_IMAGEMERGEHEIGHTTHRESHOLD should be ICAP_IMAGEMERGEHEIGHTTHRESHOLD

CAP_SUPPORTEDEXTIMAGEINFO should be ICAP_SUPPORTEDEXTIMAGEINFO

Otherwise

[DROPPED, because of acceptance of previous item] Page 10-63
CAP_IMAGEMERGEHEIGHTTHRESHOLD, ICAP_IMAGEMERGE Should be CAP_IMAGEMERGE

[DROPPED, we don't want to encourage pick-and-choose labeling] Should these be returned by MSG_QUERYSUPPORT or does the just try and fail.

Add for MSG_QUERYSUPPORT

#define TWQC_GETHELP 0x0020

#define TWQC_GETLABEL 0x0040

#define TWQC_GETGETLABELENUM 0x0080

These items are for small corrections in the spec.

[DONE] page A-25 ICAP_FEEDERENABLED should be CAP_FEEDERENABLED

[DONE] page A34 ICAP_AUTOBORDERDETECTION Should be ICAP_AUTOMATICBORDERDETECTION

[DONE] Page 4-19 Disk File Mode Transfer, remove or DG_CONTROL /DAT_SETUPFILEXFER2

During States 4, 5, or 6:

To set up the transfer the DG_CONTROL / DAT_SETUPFILEXFER or DG_CONTROL / DAT_SETUPFILEXFER2 operations of MSG_GET, MSG_GETDEFAULT, and MSG_SET can be used.

[DONE] Page 5-13 Disk File Mode Transfer, MSG_SET instead of MSG_GET in two places.

The application may determine all of the Source's supported file formats by using the ICAP_IMAGEFILEFORMAT capability. Based on this information, the application can request a particular file format and define its own choice of file name for the transfer. The desired file format and file name will be communicated to the Source in a DG_CONTROL / DAT_SETUPFILEXFER / MSG_SET.

When the Source receives the DG_IMAGE / DAT_IMAGEFILEXFER / MSG_SET operation, it should transfer the data into the designated file. The following conditions may exist:

[DONE] Page 5-14 Buffered Memory mode Transfer, MSG_SET instead of MSG_GET in one places.

When an application issues a DG_IMAGE / DAT_IMAGEMEMXFER / MSG_SET operation, check the TW_IMAGEMEMXFER.Memory.Length field to determine the size of the buffer being presented to you. If it does not fit the recommendations, fail the operation with TWRC_FAILURE / TWCC_BADVALUE

[DONE] page 10-61 CAP_IMAGEMERGE, TWIMFRONTONBOTTOM Should be TWIM_FRONTONBOTTOM and TWIMFRONTONRIGHT Should be TWIM_FRONTONRIGHT

[DONE] Page 10-73 CAP_PAPERDETECTABLE, from container for MSG_SET remove TW_ENUMERATION // 2.0 and higher

Container for MSG_SET: MSG_SET not allowed

[DONE] Page 10-89 CAP_SEGMENTED, TWSG_DISABLED should be TWSG_NONE and add to Allowed Values TWSG_AUTO

[DONE] Page 8-68 DAT_ TWUNKIDENTITY 0x000B Should be removed from here and added to the deprecated section on page 8-100

[DONE] Page 10-132 ICAP_EXTIMAGEINFO Add to See Also ICAP_SUPPORTEDEXTIMAGEINFO

[DONE] Page 7-114 DG_IMAGE / DAT_EXTIMAGEINFO / MSG_GET Add to See Also ICAP_SUPPORTEDEXTIMAGEINFO

[DONE, can't find a problem] Page 10-89 ICAP_AUTOBORDERDETECTION Should be

[Deferred to TWAIN 2.2]

For 2.2

DAT_IDENTITY / MSG_GET and MSG_GETCURRENT

DAT_IDENTITY / MSG_GETCURRENT is not documented or mentioned anywhere.

DAT_IDENTITY / MSG_GET is consumed by the DS and not the DSM. If an application is going to send this command in state 3 to be consumed by the DSM then it should be documented. Added to 2.2.

For the ToDo list:

Add to FAQ – "How to Create a GUID" to fulfill Spec CAP_CUSTOMINTERFACEGUID

The Source writer is responsible for creating a GUID. This GUID guarantees that the custom numeric values have exactly the same meaning for any Source that reports that GUID. If you need to create a GUID, but don't know how, go to the TWAIN Working Group website for help.

DSM – Sources are supposed to have an icon resource ID (page 5-2). Update the DSM select source dialog to display this icon beside the names listed.

New defines for constants in Chap 8 waiting for ok on top items. Updated twain.h with tentative defines in svn