TWAIN Errata

For Version 2.2

March 12th, 2013



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
January 17 th , 2012	Initial version
May 31st, 2012	Incorporated new content.
June 12th, 2012	Started organizing the document
July 31st, 2012	More work organizing
August 6th, 2012	More stuff, including Mac OS X
August 12th, 2012	Changes for Quarterly Tech meeting
September 4 th , 2012	Updated from the August meeting, dropped in additional items
	from Sarah and Spike
October 2 nd , 2012	More updates
October 3 rd , 2012	Updates after the Tech call
October 16 th , 2012	Fixes from Sarah, work on the "defaults"
October 17 th , 2012	Updates after the Tech call
October 23 rd , 2012	Updates after the Tech call
November 6 th , 2012	Updates before quarterly meeting
November 7 th , 2012	Locked down, new stuff will go into Errata for TWAIN 2.3
January 6 th , 2013	Corrections to the doc, no other changes
March 5, 2013	Hid technical notes and removed items
March 12, 2013	Hid last chapter, removed change bars

Contents

Fix All TW_ARRAY Capabilities [READY]	5
TW_ENUMERATION [READY]	8
CAP_EXTENDEDCAPS [READY]	9
TWCC_CAPSEQERROR [READY]	12
CAP_ENDORSER vs CAP_PRINTERINDEX [READY]	13
MSG_SETCONSTRAINT [READY]	14
CAP_AUTOFEED, CAP_CLEARPAGE, CAP_FEEDERLOADED, CAP_FEEDPAGE and CAP_REWINDPAGE [READY]	15
MSG_QUERYSUPPORTED and TWCC_CAPSEQERROR [READY]	17
CAP_FEEDERALIGNMENT [READY]	18
CAP_LANGUAGE [READY]	19
CAP_UICONTROLLABLE [READY]	20
ICAP_COMPRESSION [READY]	21
ICAP_PHYSICALHEIGHT [READY]	22
ICAP_PHYSICALWIDTH [READY]	23
ICAP_SUPPORTEDSIZES [READY]	24
ICAP_XFERMECH [READY]	25
TW_STR1024 [READY]	26
MSG_SETCONTRAINT Missing From Table [READY]	27
Legacy Issues [READY]	28
ICAP_BITDEPTH [READY]	29
ICAP_BITORDER [READY]	29
ICAP_PIXELFLAVOR [READY]	30
ICAP_XFERMECH [READY]	31
ICAP_XNATIVERESOLUTION and ICAP_YNATIVERESOLTION [READY]	32
MSG_CONSTRAINABLE Needs To Be Removed [READY]	33
Extended Capabilities [READY]	34
Add section for MSG_SETCONSTRAINT [READY]	35
Fix Sections Describing Constraints [READY]	36
Mac OS X Changes [READY]	39
TWAIN.H [READY]	45
Typos [READY]	46
Self-Cert off the TWAIN website [READY]	49
Addition to MSG_GETDEFAULTS Description [REMOVED]	49
Fixes For TW_ENUMERATION CAPABILITIES [REMOVED]	50
CAP_DEVICEONLINE [REMOVED]	50

CAP_EXTENDEDCAPS [REMOVED]	50
General Comments [DONE]	51

Fix All TW_ARRAY Capabilities [READY]

MSG_SETCONSTRAINT is used to limit the choices presented to the user on the GUI. MSG_SET is intended to select the current values.

[Update Instructions]

Page 10-13 (PDF page 419) add 'TW_ARRAY' to CAP_ALARMS...

Containers

MSG_SET: TW_ONEVALUE

TW_ARRAY

[Update Instructions]

Page 10-27 (PDF page 433) add 'TW_ARRAY' to CAP_CAMERAORDER...

Containers

MSG_SET: TW_ONEVALUE

TW_ARRAY

[Update Instructions]

Page 10-40 (PDF page 446) add 'TW_ARRAY' to CAP_DEVICEEVENT...

Containers

MSG_SET: TW_ONEVALUE

TW_ARRAY

[Update Instructions]

Page 10-43 (PDF page 449) add 'TW_ARRAY' to CAP_DOUBLEFEEDDETECTION...

Containers

MSG_SET:

TW_ONEVALUE

TW_ARRAY

[Update Instructions]

Page 10-45 (PDF page 451) add 'TW_ARRAY' to CAP_DOUBLEFEEDDETECTIONRESPONSE...

Containers

... MSG SET:

TW_ONEVALUE

TW ARRAY

[Update Instructions]

Page 10-52 (PDF page 458) add 'TW_ARRAY' to CAP_EXTENDEDCAPS...

Containers

MSG_SET: TW_ONEVALUE

TW_ARRAY

[Update Instructions]

Page 10-58 (PDF page 464) add 'TW_ARRAY' to CAP_FEEDERPOCKET...

Containers

... MSG_SET:

TW_ONEVALUE

TW_ARRAY

[Update Instructions]

Page 10-63 (PDF page 469) add 'TW_ARRAY' to CAP_INDICATORSMODE...

Containers

MSG_SET:

TW ONEVALUE

TW_ARRAY

[Update Instructions]

Page 10-72 (PDF page 478) add 'TW_ARRAY' to CAP_PAPERHANDLING...

Containers

MSG_SET: TW_ONEVALUE

TW_ARRAY

[Update Instructions]

Page 10-116 (PDF page 522) add 'TW_ARRAY' to ICAP_BARCODESEARCHPRIORITIES...

Containers

MSG_SET:

TW_ONEVALUE

TW_ARRAY

[Update Instructions]

Page 10-130 (PDF page 536) add 'TW_ARRAY' to ICAP_CUSTHALFTONE...

Containers

... MSG_SET:

TW_ONEVALUE

TW ARRAY

[Update Instructions]

Page 10-136 (PDF page 542) add 'TW_ARRAY' to ICAP_FILTER...

Containers

MSG_SET:

TW_ONEVALUE

TW ARRAY

[Update Instructions]

Page 10-147 (PDF page 553) add 'TW_ARRAY' and 'TW_RANGE' to ICAP_IMAGEDATASET...

Containers

MSG_SET: TW_ONEVALUE (see note below)

TW_ARRAY (see note below)
TW_RANGE (see note below)

[Update Instructions]

Page 10-175 (PDF page 581) add 'TW_ARRAY' to ICAP_PATCHCODESEARCHPRIORITIES...

Containers

...
MSG_SET: TW_ONEVALUE

TW_ARRAY

TW_ENUMERATION [READY]

[Update Instructions]

Page 8-22 (PDF page 310), Replace the first paragraph under the Description with the text in red...

Description

An enumeration stores a list of individual values, with one of the items designated as the current value.

There is no required order to the values in the list. However, it is recommended that the data source's GUI show the values in the order that they have been negotiated by the application.

It is also recommended, but not required, that a MSG_GET operation reflects the same order as the last MSG_SET operation for that capability.

Data sources may opt to always order some enumerated lists, like ICAP_XRESOLUTION, so that the values are presented on the GUI in numerical order.

This structure is related in function and purpose to TW_ARRAY, TW_ONEVALUE and TW_RANGE.

CAP_EXTENDEDCAPS [READY]

[Update Instructions]

Page 4-15 (PDF page 87), change the content in red (middle of the page)...

during State 4, that a particular capability be set later (during States 5, 6 or 7).

[Update Instructions]

Page 4-15 (PDF page 87), change the content in red (near the bottom of the page)...

MSG_GET

Indicates the capabilities the Source is willing to negotiate in States 5, 6 or 7.

MSG_SET

Specifies which capabilities the application wishes to negotiate in States 5, 6 or 7. For TWAIN 2.3 or later data sources, this value will already to be set to the values allowed by the data source, the list never starts empty.

MSG_GETCURRENT

Provides an array of the capabilities the Source allows to be negotiated in States 5, 6 and 7. For TWAIN 2.3 or later data sources, this value will already to be set to the values allowed by the data source, the list never starts empty.

[Update Instructions]

Page 4-15 (PDF page 87), change the content in red (bottom of the page)...

If an application attempts to set a capability in State 5, 6 or 7 and the Source has not previously agreed to this arrangement, then operation will fail with a Return Code of TWRC_FAILURE and a Condition Code of TWCC_SEQERROR.

[Update Instructions]

Page 4-30 (PDF page 102), change the content in red (middle of the page)...

In addition to automatic document feeding, TWAIN provides an option for an application to manually control the feeding of documents. This is only possible if the Source agrees to negotiate the following capabilities during States 5, 6 and 7, as indicated by CAP_EXTENDEDCAPS.

[Update Instructions]

Page 5-6 (PDF page 118), change the content in red (middle of the page)...

Once the Source in enabled, the application may only inquire about capabilities. An attempt to set a capability fails with TWRC_FAILURE / TWCC_SEQERROR, unless allowed by the CAP_EXTENDEDCAPS capability.

[Update Instructions]

Page 7-28 (PDF page 170), change the content in red (near the top of the page)...

Valid States

4 (when indicated by MSG_QUERYSUPPORT) 5, 6, 7 (when the capability appears in the CAP_EXTENDEDCAPS array, and when indicated by MSG_QUERYSUPPORT))

[Update Instructions]

Page 7-28 (PDF page 170), change the content in red (middle of the page)...

Note that this operation is only valid in State 4, unless permitted by the presence of the capability in the CAP_EXTENDEDCAPS array.

[Update Instructions]

Page 7-32 (PDF page 174), change the content in red (near the top of the page)...

Valid States

4 (when indicated by MSG_QUERYSUPPORT) 5, 6, 7 (when the capability appears in the CAP_EXTENDEDCAPS array, and when indicated by MSG_QUERYSUPPORT))

[Update Instructions]

Page 7-33 (PDF page 175), change the content in red (near the bottom of the page)...

Return TWRC_FAILURE / TWRC_SEQERROR

• If the application sends MSG_SET in State 5, 6 or 7 and the capability is not allowed by CAP_EXTENDEDCAPS.

[Update Instructions]

Page 7-35 (PDF page 177), change the content in red (near the top of the page)...

Valid States

4 (when indicated by MSG_QUERYSUPPORT) 5, 6, 7 (when the capability appears in the CAP_EXTENDEDCAPS array, and when indicated by MSG_QUERYSUPPORT))

[Update Instructions]

Page 7-36 (PDF page 178), change the content in red (near the top of the page)...

Return TWRC_FAILURE / TWRC_SEQERROR

• If the application sends MSG_SETCONSTRAINT in State 5, 6 or 7 and the capability is not allowed by CAP EXTENDEDCAPS.

[Update Instructions]

Page 10-4 (PDF page 410), change the content in red (middle of the page)...

CAP_EXTENDEDCAPS

Capabilities negotiated in States 5, 6 and 7

[Update Instructions]

Page 10-52 (PDF page 458), replace the entire 'Application' section (near the top of the page)...

Application

MSG_GET and MSG_GETCURRENT return an array of the capabilities the Source supports in States 5, 6 and 7. If either the Source or the application is older than TWAIN 2.3, use MSG_GET to get the list of allowed capabilities, and MSG_GETCURRENT to check the capabilities currently set.

MSG_SET is only needed with Sources older than TWAIN 2.3, to set the capabilities the application wants to negotiate in States 5, 6 and 7.

Stated another way, beginning with TWAIN 2.3 CAP_EXTENDEDCAPS works more like CAP_SUPPORTEDCAPS; it should be treated as a read only array, but data sources must still permit MSG_SET and MSG_RESET operations for legacy applications.

TWCC_CAPSEQERROR [READY]

```
[Update Instructions]
Page 7-14 (PDF page 156), add the content in red...
      TWCC CAPSEQERROR
                         /* Capability has a dependency on another */
                          /* capability. Sources 1.6 and newer must */
                          /* use this instead of using TWCC_BADCAP
[Update Instructions]
Page 7-17 (PDF page 159), add the content in red...
                          /* Capability has a dependency on another
      TWCC_CAPSEQERROR
                          /* capability. Sources 1.6 and newer must */
                          /* use this instead of using TWCC_BADCAP
[Update Instructions]
Page 7-20 (PDF page 162), add the content in red...
      TWCC_CAPSEQERROR
                          /* Capability has a dependency on another
                          /* capability. Sources 1.6 and newer must */
                          /* use this instead of using TWCC_BADCAP
[Update Instructions]
Page 7-28 (PDF page 170), add the content in red...
      TWCC CAPSEQERROR
                         /* Capability has a dependency on another */
                          /* capability. Sources 1.6 and newer must */
                          /* use this instead of using TWCC_BADCAP
[Update Instructions]
Page 7-31 (PDF page 173), add the content in red...
      TWCC_CAPSEQERROR
                         /* Capability has a dependency on another
                          /* capability. Sources 1.6 and newer must */
                          /* use this instead of using TWCC_BADCAP
[Update Instructions]
```

Pages 7-30 and 7-31(PDF page 172 and 173), add fix the formatting of the /* comments (they don't line up

properly).

CAP_ENDORSER vs CAP_PRINTERINDEX [READY]

[Update Instructions]

Page 3-40 (PDF page 72), add the content between the CAP_DUPLEXENABLED section and the ICAP_FRAMES section.

CAP_ENDORSER vs CAP_PRINTERINDEX

Technically, endorsers differ from printers. Printers are typically used to mark physical sheets so that it's easier to correlate images with physical documents. Endorsers are used to confirm that a given sheet of paper has passed through the scanner, usually with some kind of non-ink stamp.

True endorsers are rare, and have been used interchangeably with printers. TWAIN applications and data sources should treat them as identical.

Data Sources

Deprecate the use of CAP_ENDORSER in favor of CAP_PRINTER, which offers more options. If there's a history of using CAP_ENDORSER, map it to CAP_PRINTERINDEX.

Applications

Check for CAP_PRINTERINDEX, and use it when it's available. Be prepared to check for CAP_ENDORSER with pre-TWAIN 2.3 data sources.

[Update Instructions]

Page 10-51 (PDF page 457), replace the "Description" content with the following...

Description

Allows the application to specify the scanner's starting endorser / imprinter number.

When available, use CAP_PRINTERINDEX, instead. See the Legacy Issues section on CAP_ENDORSER vs CAP_PRINTER for more information.

[Update Instructions]

Page 10-51 (PDF page 457), same page as above, replace the "See Also" content with the following

See Also

Best Practices
CAP PRINTERINDEX

CAP_AUTOFEED, CAP_CLEARPAGE, CAP_FEEDERLOADED, CAP_FEEDPAGE and CAP_REWINDPAGE [READY]

[Update Instructions]

Page 10-16 (PDF page 422), change 'TWCC_CAPUNSUPPORTED' to 'TWCC_CAPSEQERROR' for CAP AUTOFEED...

Source

If CAP_FEEDERENABLED equals FALSE, return TWRC_FAILURE / TWCC_CAPSEQERROR (capability not supported in the current settings).

[Update Instructions]

Page 10-34 (PDF page 440), change 'TWCC_CAPUNSUPPORTED' to 'TWCC_CAPSEQERROR' for CAP CLEARPAGE...

Source

If CAP_FEEDERENABLED equals FALSE, return TWRC_FAILURE / TWCC_CAPSEQERROR (capability not supported in the current settings).

[Update Instructions]

Page 10-56 (PDF page 462), change 'TWCC_CAPUNSUPPORTED' to 'TWCC_CAPSEQERROR' for CAP_FEEDERLOADED...

Source

If CAP_FEEDERENABLED equals FALSE, return TWRC_FAILURE / TWCC_CAPSEQERROR (capability not supported in the current settings).

[Update Instructions]

Page 10-60 (PDF page 466), change 'TWCC_CAPUNSUPPORTED' to 'TWCC_CAPSEQERROR' for CAP FEEDPAGE...

Source

If CAP_FEEDERENABLED equals FALSE, return TWRC_FAILURE / TWCC_CAPSEQERROR (capability not supported in the current settings).

[Update Instructions]

Page 10-86 (PDF page 492), change 'TWCC_CAPUNSUPPORTED' to 'TWCC_CAPSEQERROR' for CAP REWINDPAGE...

Source

If CAP_FEEDERENABLED equals FALSE, return TWRC_FAILURE / TWCC_CAPSEQERROR (capability not supported in the current settings).

MSG_QUERYSUPPORTED and TWCC_CAPSEQERROR [READY]

[Update Instructions]

Page 7-51 (PDF page 168), add the content in red...

Source

If the application requests this operation on a capability your Source does not recognize (and you're sure you've implemented all the capabilities that you're required to), do not disregard the operation, but fill out the TWON_ONEVALUE container with a value of zero(0) for the Item field, indicating no support for any of the DAT CAPABILITY operations, and return a status of TWRC_SUCCESS.

If the capability will currently return TWRC_FAILURE / TWCC_CAPSEQERROR, because its availability depends on that of other capabilities, then fill out the TWON_ONEVALUE container with a value of zero (0) for the Item field, indicating no support for any of the DAT CAPABILITY operations, and return a status of TWRC_SUCCESS.

This is a memory allocation operation. It is possible for this operation to fail due to a low memory condition. Be sure to verify that the allocation is successful. If it is not, attempt to reduce the amount of memory occupied by the Source. If the allocation cannot be made return TWRC_FAILURE with TWCC_LOWMEMORY to the application and set the pCapability->hContainer handle to NULL.

CAP_FEEDERALIGNMENT [READY]

[Update Instructions]

Page 10-53 (PDF page 459), add the text marked in red...

Containers

MSG_GET TW_ONEVALUE

TW_ENUMERATION

MSG_GETDEFAULT RW_ONEVALUE

CAP_LANGUAGE [READY]

[Update Instructions]

Page 10-66 (PDF page 473), add the text marked in red ('_USA')...

// 1.8 should use these...

TWLG_ENGLISH_AUSTRALIAN
TWLG_ENGLISH_CANADIAN
TWLG_ENGLISH_IRELAND
TWLG_ENGLISH_NEWZEALAND
TWLG_ENGLISH_SOUTHAFRICA
TWLG_ENGLISH_UK
TWLG_ENGLISH_USA
TWLG_ESTONIAN

CAP_UICONTROLLABLE [READY]

[Update Instructions]

Page 5-14 (PDF page 126), change the content in red...

CAP_DEVICEONLINE	MSG_GET required	1.6
CAP_UICONTROLLABLE	MSG_GET required	1.6

...

CAP_UICONTROLLABLE (Value = TRUE) 1.9

[Update Instructions]

Page 10-98 (PDF page 504), change the content in red...

Source

This capability was introduced in TWAIN 1.6. All Sources compliant with TWAIN 1.6 and above must support this capability. Sources that are not TWAIN 1.6-compliant may return TWRC_FAILURE / TWCC_BADCAP if they do not support this capability.

All Sources compliant with TWAIN 1.9 and above must support the ability to scan without the UI (TW_USERINTERFACE.ShowUI = 0 and CAP_INDICATORS = FALSE), therefore they must report a value of TRUE for this capability.

ICAP_PHYSICALHEIGHT [READY]

[Update Instructions]
Page 10-177 (PDF page 583), replace this...

Source

For a flatbed scanner, the scannable height of the platen. For a handheld scanner, the maximum length of a scan.

...with this...

Source

For a flatbed scanner, the height of the platen; for a handheld scanner or a sheet fed scanner, the maximum length of a scan.

ICAP_PHYSICALWIDTH [READY]

[Update Instructions]

Page 10-178 (PDF page 584), replace this...

Source

For a flatbed scanner, the scannable width of the platen. For a handheld scanner, the maximum width of a scan.

...with this...

Source

For a flatbed scanner, the width of the platen; for a handheld scanner or a sheet fed scanner, the maximum width of a scan.

ICAP_SUPPORTEDSIZES [READY]

[Update Instructions]

Page 10-190 (PDF page 596), change '*TWSS_A6' to TWSS_A6'...

Values

...

Allowed Values: .

. . .

TWSS_USLEGAL TWSS_A6

[Update Instructions]

Page 10-191 (PDF page 597), add closing parenthesis for TWSS_A4LETTER note in first column...

TWSS_A4(TWSS_A4LETTER) TWSS_JISB6

ICAP_XFERMECH [READY]

[Update Instructions]

Page 10-197 (PDF page 603), add the items in red...

See Also

```
Best Practices
DG_IMAGE / DAT_IMAGEFILEXFER / MSG_GET
DG_IMAGE / DAT_IMAGEMEMFILEXFER / MSG_GET
DG_IMAGE / DAT_IMAGEMEMXFER / MSG_GET
DG_IMAGE / DAT_IMAGENATIVEXFER / MSG_GET
```

TW_STR1024 [READY]

[Update Instructions]
Page 8-97 (PDF page 386), add ", FAR *pTW_STR1024"...

String types

typedef unsigned char TW_STR1024[1026], FAR *pTW_STR1026, FAR *pTW_STR1024; typedef wchar_t TW_UNI512[512], FAR *pTW_UNI512;

MSG_SETCONTRAINT Missing From Table [READY]

[Update Instructions]

Page 7-2 (PDF page 144), add MSG_SETCONTRAINT with the appropriate page number...

	MSG_RESETALL	7-30
	MSG_SET	7-32
	MSG_SETCONSTRAINT	7-3?
DG_CONTROL DAT_CUSTOMDSDATA	MSG_GET	7-38
	MSG_SET	7-39

Legacy Issues [READY]

[Update Instructions]

Page 3-39 (PDF page 71), add the text in red...

ICAP_BITDEPTH

Data Sources

Report the number-of-channels times the depth-per-channel. For example, a typical value for ICAP_BITDEPTH when ICAP_PIXELTYPE is TWPT_RGB is 3 \times 8 = 24.

Applications

Ambiguity in the Specification prior to version 2.2 may result in some Data Sources reporting just the depth-per-channel. In the majority of cases a value of 8 for ICAP_BITDEPTH when ICAP_PIXELTYPE is TWPT_RGB may be treated as if the bit depth is really 24.

Also, owing to a bug in an old version of the sample driver, some Data Sources may report all of their possible bit depth values, instead of those that apply just to the current ICAP_PIXELTYPE value. For instance, with a setting of TWPT_RGB ICAP_BITDEPTH may report allowed value of 1, 8 and 24, when only 24 is really permitted.

[Update Instructions] Page 3-40 (PDF page 72), add the text in red...

ICAP_FRAMES

Applications

Some scanners may handle having the origin of a frame as 0,0 differently. The spec states that when an application is only interested in the extent of image scanned it can set the origin to 0,0 with MSG_SET. Some center feed or right feed scanners may scan from the left edge of the scanner. They expect the application to center (or right align) the frame using the physical extent of the scanner.

ICAP_XFERMECH

Data Sources

Applications are supposed to alert a data source to the transfer mechanism they'll be using in states 6 and 7 by setting ICAP_XFERMECH. However, not all applications do this. So, when possible, a data source should tolerate this, and use return the image data using whatever DAT_IMAGE*XFER call the application selects.

ICAP_XFERMECH [READY]

[Update Instructions]

Page 10-197 (PDF page 603), fix the red text

Error

A TWAIN driver must support TWSX_MEMORY and TWSX_NATIVE, therefore it must use TW_ENUMERATION when ICAP_XFERMECH is queried using MSG_GET. Applications must still be prepared to handle TW_ONEVALUE returns; however any TWAIN 2.2 driver that does this has not passed TWAIN Certification.

Instructions for change

Add the red text.

Containers

MSG_GET	TW_ONEVALUE (permitted TWAIN 2.1 and earlier)
	TW_ENUMERATION (required TWAIN 2.2 and later)
MSG_GETCURRENT	TW_ONEVALUE
MSG_GETDEFAULT	TW_ONEVALUE
MSG_SET	TW_ONEVALUE
MSG_SETCONSTRAINT	TW_ONEVALUE
	TW_ENUMERATION
MSG_RESET	TW_ONEVALUE
MSG_QUERYSUPPORT	TW ONEVALUE

MSG_CONSTRAINABLE Needs To Be Removed [READY]

[Update Instructions]

Page 7-25 (PDF page 167), remove the item in red that's been struck through...

Source

...

2. Item = Bit pattern representing the set of operation that are supported by the Data Source on this capability (TWQC_GET, TWQC_SET, TWQC_GETCURRENT, TWQC_GETDEFAULT, TWQC_RESET, TWQC_SETCONSTRAINT, TWQC_CONSTRAINABLE);

[Update Instructions]

Page 8-66 (PDF page 354), remove the item in red that's been struck through...

1.6	TWQC_RESET	0x0010
2.2	TWQC_SETCONSTRAINT	0×0020
2.2	TWQC_CONSTRAINABLE	0x0040
	TWOC GETHELP	0×0100

[Update Instructions]

Page 8-97 (PDF page 387), insert the item in red (make sure there are blank lines before and after it)...

Messages MSG_INVOKE_CALLBACK 0x0903 MSG_CHECKSTATUS 0x0201

Query Support TWQC_CONSTRAINABLE 0x0040

Capability values TWSX_FILE2

Extended Capabilities [READY]

[Update Instructions] Page A-32 (PDF page 696), replace this...

Extended Capabilities

This is an appropriate times to negotiate the extended capabilities CAP_EXTENDEDCAPS as indicated by CAP_SUPPORTEDEXTCAPS (ones that are settable in state 6), though there is no implied dependency in locating it here.

...with this...

Extended Capabilities

Beginning with TWAIN 2.3 the Data Source always sets CAP_EXTENDED CAPS to the array of capabilities that are negotiable in States 5, 6 and 7. The application reads this array, or (for legacy purposes) it can set the array to the desired values, and, if TWRC_CHECKSTATUS is returned, follow up to see which values were accepted.

Add section for MSG_SETCONSTRAINT [READY]

[Update Instructions]

Page A-44 (PDF page 708), replace this...

MSG_SET sets the current value and optionally sets the constraints on a capability. Sources must never save
the constraints negotiated by an application. The case of the current value is a little different, since a Source
is supposed to reflect the negotiated values in its UI, it's possible for a capability set in State 4 to find its way
into the user defaults.

...with this...

- MSG_SET sets the current value. Therefore, it's possible for a capability set in State 4 to find its way into the
 user defaults.
- MSG_SETCONSTRAINT sets the current value and optionally sets the constraints on a capability. Sources
 must never save the constraints negotiated by an application. However, it's possible for a capability set in
 State 4 to find its way into the user defaults.

Fix Sections Describing Constraints [READY]

[Update Instructions]

Page 3-35 (PDF page 67), add the line in red...

If setting constraints, then do the following:

- call DG_CONTROL / DAT_CAPABILITY / MSG_QUERYSUPPORT to confirm that the capability has TWQC_SETCONSTRAINT
- call DG_CONTROL / DAT_CAPABILITY / MSG_GET on the desired capability
- determine the container type from the TW_CAPABILITY.ConType field

[Update Instructions]

Page 3-36 (PDF page 68), add the text in red...

- free the original container using the DAT_ENTRYPOINT.DSM_MemFree function
- call MSG_SETCONSTRAINT with the updated container
- free the container using the DAT_ENTRYPOINT.DSM_MemFree function
- respond to the status returned by MSG_SETCONSTRAINT

[Update Instructions]

Page A 44 (PDF page 708), add the text in red (note the missing 't' in 'constraints')...

• MSG_SETCONSTRAINT sets the current value and optionally sets the constraints on a capability. Sources must never save the constraints negotiated by an Application. The case of the current value is a little different, since a Source is supposed to reflect the negotiated values in its UI, it's possible for a capability set in State 4 to find its way into the user defaults.

[Update Instructions]

Page 3-17 (PDF page 49), remove the red text that is struck out...

MSG SET

Changes the Current Value(s) of the specified capability to those requested by the application.

Note: Source is not required to limit values based on the application's request although it is strongly recommended that they do so.

Page 3-17 (PDF page 49), add the text in read between the two bits in black...

Note: It is important here to once again remind application writers to always check the return code from any negotiated capabilities transactions.

MSG SETCONSTRAINT

Changes the Current Value(s) of the specified capability to those requested by the application, and constrains the allowable contents of an array, enumeration or range container.

If the Return Code indicates TWRC_FAILURE, check the Condition Code. A code of TWCC_BADVALUE can mean:

- The application sent an invalid value for this Source's container.
- The Source doesn't allow the type of container used by the application to set this capability.

Capability negotiation gives the application developer power to guide the Source and control the images they receive from the Source. The negotiation typically occurs during State 4. The following material illustrates only one very basic capability and container structure. Refer to Chapter 4, "Advanced Application Implementation" for a more extensive discussion of capabilities including information on how to delay the negotiation of some capabilities beyond State 4.

Note: It is important here to once again remind application writers to always check the return code from any negotiated capability transactions.

Set the Capability to Specify the Number of Images the Application can Transfer

[Update Instructions]

Page 4-4 (PDF page 49), add the text in red...

To limit the Available Values:

Use DG_CONTROL / DAT_CAPABILITY / MSG_SETCONSTRAINT and one of the following containers:

Page 4-8 (PDF page 80), add the text in red...

MSG_SET (applies if either the application or the driver is TWAIN 2.1 or less)

As indicated in Chapter 7, "Operation Triplets", description of this capability triplet: "Current Values are set when the container is a TW_ONEVALUE or TW_ARRAY. Available and Current Values are set when the container is a TW_ENUMERATION or TW_RANGE."

To further clarify this operation, it should be stated that when an application imposes a constraint, the data source must consider the set of supported values and the set of requested constraints. The resulting set of values shall contain only the values that are shared by those supported and those requested.

A condition may arise after constraints are imposed, where the default value is no longer within the set of supported values. When using a TW_ENUMERATION, the reported default index should be changed by the data source to something that falls within the new constrained set. This is simply a precaution to ensure it is a valid index. In this case, the Default index in a TW_ENUMERATION loses meaning and should be ignored by applications, since MSG_RESET shall cause the constraints to be eliminated.

MSG_SET (applies if both the application and the driver is TWAIN 2.2 or more)

When both the application and the driver are TWAIN 2.2 or higher MSG_SET only changes the current value, it has no effect on the available values. This applies regardless of the container type used. In other words, TW_ENUMERATION and TW_RANGE can be used to set the current value using MSG_SET. In the case of TW_ENUMERATION only the ItemType, CurrentIndex and ItemList fields are used to get the current value. In the case of TW_RANGE only the ItemType and CurrentValue fields are used.

MSG SETCONSTRAINT (applies if both the application and the driver is TWAIN 2.2 or more)

As indicated in Chapter 7, "Operation Triplets", description of this capability triplet: "Current Values are set when the container is a TW_ONEVALUE or TW_ARRAY. Available and Current Values are set when the container is a TW_ENUMERATION or TW_RANGE."

To further clarify this operation, it should be stated that when an application imposes a constraint, the data source must consider the set of supported values and the set of requested constraints. The resulting set of values shall contain only the values that are shared by those supported and those requested.

A condition may arise after constraints are imposed, where the default value is no longer within the set of supported values. When using a TW_ENUMERATION, the reported default index should be changed by the data source to something that falls within the new constrained set. This is simply a precaution to ensure it is a valid index. In this case, the Default index in a TW_ENUMERATION loses meaning and should be ignored by applications, since MSG_RESET shall cause the constraints to be eliminated.

Mac OS X Changes [READY]

```
[Update Instructions]
Page 8-6 (PDF page 294), replace this content...
      #ifdef TWH_CMP_MSC
                    #pragma pack (push, before_twain)
                    #pragma pack (2)
             #elif TWH_CMP_GNU
                    #pragma pack (push, before_twain)
                    #pragma pack (2)
             #elif TWH_CMP_BORLAND
                    #pragma option -a2
      #elif TWH_CMP_XCODE
             #if PRAGMA STRUCT ALIGN
                    #pragma options align=mac68k
             #elif PRAGMA_STRUCT_PACKPUSH
                    #pragma pack (push, 2)
             #elif PRAGMA_STRUCT_PACK
                    #pragma pack (2)
             #endif
      #elif
...with this...
      #ifdef TWH_CMP_MSC
             #pragma pack (push, before_twain)
             #pragma pack (2)
      #elif defined(TWH_CMP_GNU)
             #if defined(__APPLE__) /* cf: Mac version of TWAIN.h */
                    #pragma options align = power
             #else
                    #pragma pack (push, before_twain)
                    #pragma pack (2)
             #endif
      #elif defined(TWH_CMP_BORLAND)
             #pragma option -a2
      #endif
```

Page 8-6 (PDF page 294), replace this content...

```
/* Restore the previous packing alignment: this occurs after all
      structures are defined */
      #ifdef TWH CMP MSC
                    #pragma pack (pop, before_twain)
             #elif TWH CMP GNUC
                    #pragma pack (pop, before_twain)
             #elif TWH_CMP_BORLANDC
                    #pragma option -a.
      #elif TWH_CMP_XCODE
             #if PRAGMA_STRUCT_ALIGN
                    #pragma options align=reset
             #elif PRAGMA STRUCT PACKPUSH
                    #pragma pack (pop)
             #elif PRAGMA STRUCT PACK
                    #pragma pack()
             #endif
      #endif
...with this...
      /* Restore the previous packing alignment: this occurs after all structures are defined */
      #ifdef TWH_CMP_MSC
             #pragma pack (pop, before_twain)
      #elif defined(TWH_CMP_GNU)
             #if defined(__APPLE__) /* cf: Mac version of TWAIN.h */
                    #pragma options align = reset
             #else
                    #pragma pack (pop, before_twain)
             #endif
      #elif defined(TWH_CMP_BORLAND)
             #pragma option -a.
      #endif
```

Page 8-7 (PDF page 295), replace this content...

String types

```
typedef char TW_STR32[34], FAR *pTW_STR32;
typedef char TW_STR64[66], FAR *pTW_STR64;
typedef char TW_STR128[130], FAR *pTW_STR128;
typedef char TW_STR255[256], FAR *pTW_STR255;
```

...with this...

String types

```
#if defined(_APPLE__)/* cf: Mac version of TWAIN.h */
typedef unsigned char TW_STR32[34], FAR *pTW_STR32;
typedef unsigned char TW_STR64[66], FAR *pTW_STR64;
typedef unsigned char TW_STR128[130], FAR *pTW_STR128;
typedef unsigned char TW_STR255[256], FAR *pTW_STR255;
#else

typedef char TW_STR32[34], FAR *pTW_STR32;
typedef char TW_STR64[66], FAR *pTW_STR64;
typedef char TW_STR128[130], FAR *pTW_STR128;
typedef char TW_STR255[256], FAR *pTW_STR255;
#endif
```

Page 8-7 (PDF page 295), replace this content...

```
Numeric types
```

typedef char TW_INT8, FAR *pTW_INT8; typedef short TW_INT16, FAR *pTW_INT16; typedef long TW_INT32, FAR *pTW_INT32; typedef unsigned char TW_UINT8, FAR *pTW_UINT8; typedef unsigned short TW_UINT16, FAR *pTW_UINT16; typedef unsigned long TW_UINT32, FAR *pTW_UINT32; typedef unsigned short TW_BOOL, FAR *pTW_BOOL;

...with this...

Numeric types

```
typedef char
                     TW INT8, FAR *pTW INT8;
typedef short
                     TW_INT16, FAR *pTW_INT16;
#if defined(__APPLE__) /* cf: Mac version of TWAIN.h */
       typedef int
                     TW INT32, FAR *pTW INT32;
#else
       typedef long
                     TW_INT32, FAR *pTW_INT32;
#endif
typedef unsigned char TW_UINT8, FAR *pTW_UINT8;
typedef unsigned short TW_UINT16, FAR *pTW_UINT16;
#if defined( APPLE ) /* cf: Mac version of TWAIN.h */
                            TW_UINT32, FAR *pTW_UINT32;
       typedef unsigned int
#else
       typedef unsigned long TW UINT32, FAR *pTW UINT32;
#endif
typedef unsigned short TW_BOOL, FAR *pTW_BOOL;
```

[Update Instructions]

Page 8-10 (PDF page 298), replace this content...

```
typedef struct {
      TW_MEMREF CallBackProc;
      TW_UINT32 RefCon;
      TW_INT16 Message;
} TW_CALLBACK, FAR * pTW_CALLBACK;
```

...with this...

[Update Instructions] Page 8-37 (PDF page 325), replace this content...

```
typedef struct {
                           Id;
       TW_UINT32
       TW_VERSION
                           Version;
      TW UINT16
                           ProtocolMajor;
       TW_UINT16
                           ProtocolMinor;
       TW_UINT32
                           SupportedGroups;
       TW_STR32
                           Manufacturer;
       TW_STR32
                           ProductFamily;
       TW_STR32
                           ProductName;
} TW_IDENTITY, FAR * pTW_IDENTITY;
```

...with this...

```
typedef struct {
       #if defined(__APPLE__) /* cf: Mac version of TWAIN.h */
              TW_MEMREF Id;
       #else
              TW UINT32
                           Id;
       #endif
       TW_VERSION
                            Version;
       TW_UINT16
                            ProtocolMajor;
       TW_UINT16
                            ProtocolMinor;
       TW_UINT32
                            SupportedGroups;
       TW_STR32
                            Manufacturer;
       TW_STR32
                            ProductFamily;
       TW_STR32
                            ProductName;
} TW_IDENTITY, FAR * pTW_IDENTITY;
```

```
[Update Instructions]
Page 8-54 (PDF page 342), replace this content...
       typedef struct {
              TW_UINT16 Count;
              union {
                     TW_UINT32 EOJ;
                     TW_UINT32 Reserved;
              };
      } TW_PENDINGXFERS, FAR *pTW_PENDINGXFERS;
...with this...
       typedef struct {
              TW_UINT16 Count;
              union {
                     TW_UINT32 EOJ;
                     TW_UINT32 Reserved;
                     #if defined(__APPLE__) /* cf: Mac version of TWAIN.h */
                            union {
                                   TW_UINT32 EOJ;
                                   TW_UINT32 Reserved;
                                   } TW_JOBCONTROL;
                     #endif
      } TW_PENDINGXFERS, FAR *pTW_PENDINGXFERS;
```

TWAIN.H [READY]

П	IJ	nd	ate	In	stri	act	io	nsl
Ľ	$\overline{}$	PΨ	aic	111	ou.	uci	110	110]

This fix is in TWAIN.H, no change is needed to the TWAIN Specification. Add the item in red. Be sure to leave the rest of the declaration as-is, we can't remove old content.

[Update Instructions]

This fix is in TWAIN.H, no change is needed to the TWAIN Specification. Add the item in red.

Typos [READY]

[Update Instructions]

Page 4-4 (PDF page 76), change MSG_SET to MSG_SETCONSTRAINT (about mid-way down the page).

Use DG_CONTROL / DAT_CAPABILITY / MSG_SETCONSTRAINT and one of the following containers:

[Update Instructions]

Page 4-4 (PDF page 76), add ACAP_ (near the bottom of the page).

• Cap = the CAP_, ICAP or ACAP name for the capability it is interested in

[Update Instructions]

Page 4-6 (PDF page 78), change MSG_SET to MSG_SETCONSTRAINT (top of the page).

Send the request to the Source using DG_CONTROL / DAT_CAPABILITY / MSG_SETCONSTRAINT.

[Update Instructions]

Page 7-25 (PDF page 167), add the missing 'E' in 'TW_ONEVALUE'; change 'TWTW_INT32' to 'TWTY_INT32'; and add the missing 's' to the end of 'operations'.

Fill the fields in TW ONEVALUE as follows:

- 1. ItemType = TWTY_INT32;
- 2. Item = Bit pattern representing the set of operations that are supported by the Data Source on this capability (TWQC_GET, TWQC_SET, TWQC_GETCURRENT, TWQC_GETDEFAULT, TWQC RESET, TWQC SETCONSTRAINT, TWQC CONSTRAINABLE);

[Update Instructions]

Page 10-167 (PDF page 573), change 'capacity' to 'capability'...

Source

The Source is responsible for rotating the image if it allows this capability to be set. If not supported, return TWRC_FAILURE / TWCC_CAPUNSUPPORTED.

[Update Instructions]

Page A-44 (PDF page 708), change 'MSG_QUERYINTERFACE' to 'MSG_QUERYSUPPORT'...

A simple mechanism for resetting a Source uses the following steps (*Applications that use the Source's UI should not use this method*): for each device supported by the Source (pre-1.8 Sources only have one implicit device) the Application calls CAP_SUPPORTEDCAPS; for each capability the Application calls DG_CONTROL / DAT_CAPABILITY / MSG_QUERYSUPPORT to see if it supports TWQI_RESET; if it does, then the Application sends DG_CONTROL / DAT_CAPABILITY / MSG_RESET which resets the capability.

Performing these steps will protect an Application from any user defaults created by a previous Application. Please note, not all Sources may support MSG_QUERYSUPPORT. For those that

don't, an Application would have to issue MSG_RESET on all capabilities (perhaps excluding those it knows to be read-only) and trust that the Source is robust enough to report TWRC_FAILURE for those capabilities that do not support MSG_RESET.

[Update Instructions]

Page 3-36 (PDF page 68), change 'desire' to 'desired' (about a third down the page) in the TWON_ARRAY section...

- set the .NumItems field to the number of desired elements
- set the .ItemList field with the desired values
- unlock the new container using the DAT_ENTRYPOINT.DSM_MemUnlock function

[Update Instructions]

Page 3-36 (PDF page 68), change 'desire' to 'desired' (about a two-third down the page) in the TWON ENUMERATION section...

- set the .NumItems field to the number of desired elements
- set the .ItemList field with the desired values
- unlock the new container using the DAT_ENTRYPOINT.DSM_MemUnlock function

[Update Instructions]

Page 10-104 (PDF page 510), add a comma between the words in red...

Application

ICAP_AUTOMATICCOLORENABLED must be TRUE. When it is, the Application sets this capability to specify the pixel type the Source uses when transferring non-color images.

[Update Instructions]

Page 7-27 (PDF page 169), change 'constants' to 'constraints'...

Application

• To set the Current Value of the specified capability to the Source's mandatory or preferred value, and to remove any constraints from the allowed values supported by the Source.

[Update Instructions]

Page 7-30 (PDF page 172), remove the struck-through words 'is the'...

Description

This command resets all current values back to original power-on defaults. All current values are set to their default value except is the where mandatory values are required. All constraints are removed for all of the negotiable capabilities supported by the driver.

Page 5-7 (PDF page 119), fix both words in red to be 'constraints'...

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 constraints accepts the new value and limits MaxValue to 900.0. A Source that does not support constraints 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.

Memory Allocation

[Update Instructions]

Page 11-2 (PDF page 612), change TTWRC_CANCEL to TWRC_CANCEL...

TWRC CANCEL Abort transfer or the Cancel button was pressed.

[Update Instructions]

Page A-36 (PDF page 700), change 'little' to 'no', and add the word 'but'...

Semi Independent Capabilities are small groups that have no effect on the big picture, but do have their own pockets of dependencies.

[Update Instructions]

Page 8-16 (PDF page 304), change 'based' to 'passed'... (near middle of page)

It is also recommended that source vendors embed basic source revision and vendor ID information in the hData body so they can determine if the structure being passed to the data source is correct.

[Update Instructions]

Page 8-16 (PDF page 304), change 'InfoData' to 'hData', the crossed out item was already addressed in the previous request. With regards to the "TW_UINT8 InfoData[1]; /* Array (Length) bytes long */", that item references an older version of the Spec, and so should remain as-is...

The format of the data contained in hData will be data source specific and will not be defined

information in the hData body so they can determine of the structure being passed to the data

the hData body so they can determine if the structure being passed to the data source is correct.

Self-Cert off the TWAIN website [READY]

[Update Instructions]

Page 13-3 (PDF page 631), add the text in red...

After TWAIN self-certification has been successfully completed the tester may submit an "Acknowledgement of Successful Completion of TWAIN Self-Certification" form to the TWAIN Working Group.

This can be accomplished in more than one way. The preferred method is to access the TWAIN Working Group website (www.twain.org), and access the section titled "Scanner Driver Developers." Under there is the "Certify TWAIN Driver" link.

Alternatively, one can submit a notarized or a digitally signed form of the document.

This form includes the following information