TWAIN Self-Certification Process for Data Sources

For Version 2.4

February 14th, 2018

13

TWAIN Self-Certification Process for Data Sources

Chapter Contents

The TWAIN self-certification system helps developers test their data source's support of the basic interface described by the TWAIN Specification. Passing the test helps to confirm that the data source's interface works as expected with applications, leading to a better user experience.

This document provides the Test Plan for TWAIN self-certification for data sources. It also describes how to submit a form affirming successful completion of the test to receive authorization to display the "TWAIN Certified" logo.

Overview

The TWAIN interface operates between an application and a data source. The nature of this interface is described by the TWAIN Specification.

Basic TWAIN self-certification exercises specific portions of the TWAIN interface and behavior of the TWAIN interface that all data sources are required to support. Passing these tests confirms that a data source correctly follows the TWAIN Specification, when responding to commands sent by an application, and that it does not crash or hang.

This is not a comprehensive test of the entire TWAIN interface. It focuses on enforcing basic "good behavior". More stringent tests may be described in future.

The basic self-certification test is limited to the kinds of checks described in this document. Modifications may be made in association with new versions of the TWAIN Specification (for instance, the addition of a new mandatory feature). For this reason self-certification is always done in the context of a particular version of the TWAIN Specification (ex: 2.2).

TWAIN data sources with a protocol version of 1.9 or higher may be self-certified. The version of this document is a measure of the kinds of tests performed on the data source. Running the tests in this document does not certify a TWAIN 1.9 data source as TWAIN 2.2 compliant, rather the data source is self-certified as TWAIN 1.9 compliant using criteria described inside of the TWAIN 2.2 Specification.

TWAIN data sources that have been self-certified will work correctly with any compliant TWAIN Application reporting a TWAIN protocol version of 1.5 or higher.

TWAIN self-certification promotes the creation of 64-bit applications and data sources by requiring simultaneous submissions of native 32-bit and 64-bit data sources for Windows Vista or later, Macintosh OS X or Linux. A native 64-bit data source is one that interfaces with a native 64-bit application. 64-bit applications cannot be run on 32-bit Systems. A 32-bit data source running in any kind of virtual or thunking environment on a 64-bit Operating System does not qualify as a native 64-bit data source.

TWAIN self-certification requires the presence of a TWAIN data source manager corresponding to the version of the TWAIN data source or higher. If one is not pre-installed on the operating system, then the TWAIN data source must install it.

Questions or comments regarding TWAIN self-certification should be referred to the TWAIN Forum www.twainforum.org.

Non-Goals of Basic TWAIN Self-Certification

This is a test of the operation of the interface; it does not test the internals of the data source.

This test is not designed to catch data errors (ex: bad pointers, data corruption, array out of bounds, etc) except in those instances where the error happens to cause the failure of some other test.

Negotiated settings are not confirmed in the meta-data or images they produce (ex: did changing `ICAP_BRIGHTNESS` really result in a brighter or darker image, was the proper print string written on the document).

Constraints for `TW_ENUMERATION` and `TW_RANGE` are not tested (ex: limiting the `ICAP_PIXELTYPE` enumeration to just `TWPT_RGB`, or limiting `ICAP_BRIGHTNESS` to a range of -100 to 100).

Mandatory features for accessories are not tested (ex: there is no check to make sure that all of the barcode features are properly supported if any one barcode capability is detected).

Affirmation of Successful Completion of TWAIN Self-Certification

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

Company: The name of the company manufacturing the data source being self-certified, a division within that company may be optionally provided. The submitter may also opt to provide a URL to their company's website which will link off of this name.

Hardware: The model name, model number and revision of the hardware used during self-certification. This is marketing information identifying the device supported by this specific TWAIN data source. In most cases this information can be found printed somewhere on the device.

TWAIN Data Source Identity: Fields from the TWAIN data source's `TW_IDENTITY` structure, which indicate the manufacturer, family, product, and the version number, uniquely identify the data source to the application. The `TW_IDENTITY.ProductName` should be unique by itself, since this is the only field displayed by the data source manager's user select dialog on Windows.

TWAIN Data Source Version: The complete version of the TWAIN data source, matching the .DLL version on Windows, and the .so file name on Linux and Mac OS X, this version number matches the MajorNum and MinorNum fields from the data source's `TW_IDENTITY.Version` structure.

Installation: The name and the version of the installation media that includes this TWAIN data source provides information the user needs to install the self-certified TWAIN driver.

Operating System: The operating system's name and revision (version number or service pack) that was used during self-certification.

Processor: The computer processor of the host machine used during self-certification, examples include: x86, x64, IA64. This should match the native processor supported by the TWAIN data source. For example, if the self-certification is performed for a 32-bit TWAIN data source on Windows XP or Linux without a 64-bit data source, then the x86 processor should be used.

32-Bit / 64-Bit: When performing the self-certification test on Windows Vista or later, or any version of Macintosh OS X, or Linux, the submitted form must indicate successful completion using both a native 32-bit and a native 64-bit data source.

Email: The name and email address of a contact. This is initially used to deliver the Logo, but it will also be used to help manage entries posted by the TWAIN Working Group.

URL: The URL to the Installer for the TWAIN data source is a convenience for users browsing the posted list of self-certified content. It is optional, but recommended.

Self-Certification Method: The submitter may specify the software used to perform self-certification, when indicated this information is made available to users browsing the posted list of self-certified content.

It is expected that multiple versions of the same driver will be submitted over the life of the hardware product. Please be aware of the following:

Email address: The email address specifies the contact who receives the Logo for a successful submission. This same email address must be used when submitting a new instance of a previously submitted TWAIN data source, or when replacing an existing submission. Requests using other email addresses may not be recognized by the TWAIN Working Group.

Signature: There is no requirement for the same signature (notarized or digital) to be used from one submission to the next, but pairing the same signature with the same email address for all submissions for a given driver is appreciated.

Hardware: The model name and model number uniquely identifies the hardware supported by the TWAIN data source. Submissions of new TWAIN data sources for the same hardware must take care to make sure that this information is identical from one version to the next. If there is no exact match with an existing hardware entry, then the entire entry is treated as new.

TWAIN Data Source Identity: The following fields uniquely identify the TWAIN data source: `TW_IDENTITY.Manufacturer`, `TW_IDENTITY.ProductFamily` and `TW_IDENTITY.ProductName`. When updating a previously existing self-certified TWAIN data source it is important to make sure this data is identical from one version to the next. If there is no exact match with an existing TWAIN data source, then the entire entry is treated as new.

TWAIN Data Source Version: Many vendors use a four field versioning system (ex: 1.2.0.1). The first two fields must correspond to the `TW_IDENTITY.Info.Version.Major` and `TW_IDENTITY.Info.Version.Minor` fields. The last two fields vary among vendors, and are not described here. The value zero must be used for any unused field. If a submission has exactly the same email, hardware, data source and version information as a previous submission, it will replace its posting on the TWAIN Working Group website. If there is no exact match with an existing TWAIN data source, then the entire entry is treated as new.

Operating System: The operating system's name and revision (version number or service pack) that was used during self-certification. If there is no exact match with an existing TWAIN data source, then the entire entry is treated as new.

The TWAIN Working Group makes no attempt to enforce continuity of versions. If the submission is correct, the version numbers may change in any way specified by the submitter.

Submission of the form qualifies the data source and its associated hardware to display the TWAIN Certified Logo. Submission information from the form is displayed on the TWAIN Working Group website (www.twain.org).

Contact information is required to deliver the Logo; this includes the name of a contact and an email address. This information will not be shared or made public. The form asks if the email address may be used to occasionally send information relating to TWAIN or the TWAIN Working Group.

The form must be either digitally signed or notarized. This identification is meant to guarantee that the document has not been modified since it was signed. The form includes an address where it can be mailed as a paper copy or emailed. The complete form is on the next two pages.

Form

Affirmation of Successful Completion of TWAIN Self-Certification
Compliance with TWAIN Versions 1.9 through 2.2
Page 1 of 2

Completion and submission of a digitally signed or notarized original of this statement to the TWAIN Working Group authorizes the authorized representative or their company to display the TWAIN Certified Logo on the hardware, software and marketing materials of the TWAIN data source described below. All fields must be filled in, except where otherwise indicated.

The certification mark is intended for use by authorized entities or persons and is intended to certify that this software conforms to standards designated by the TWAIN Working Group. This document indicates compliance with the TWAIN Specification for version TWAIN 2.2 or earlier.

The following information will not be published or shared. The Logo will be sent to the email address.

Name of Contact: ____

Email Address: ____

May the TWAIN Working Group send TWAIN information not related to this submissionto this email address? (circle one) [Yes] [No]

The following fields will be posted on the TWAIN Working Group website.

Company:	______________________________
Division: (optional)	______________________________
Company/Division URL: (optional)	______________________________
Hardware Model Name:	______________________________
Hardware Model Number:	______________________________
Hardware Model Revision: (optional)	______________________________
TW_IDENTITY.Manufacturer:	______________________________
TW_IDENTITY.ProductFamily:	______________________________
TW_IDENTITY.ProductName:	______________________________
TW_IDENTITY.Protocol:	______. ______
TWAIN Data Source Version:	______. ______ . ______ . ______
Installer Version:	______________________________
URL to Data Source: (optional)	______________________________

Processor: x86 ___ x64 ___ other ____________________

Operating System/Revision: ____________________

Self-Certification Software: (optional) ____________________

May the TWAIN Working Group post the software used to self-certify? (circle one)
[Yes] [No]

Affirmation of Successful Completion of TWAIN Self-Certification
Compliance with TWAIN Versions 1.9 through 2.2
Page 2 of 2

Please confirm that all tests described within the "TWAIN Self-Certification Process for Data Sources" document have been completely and successfully run (check all that apply).

32-bit	64-bit	Test
☐	☐	TWAIN Standard Capability Tests
☐	☐	Vendor Custom Capability Tests
☐	☐	Status Return Tests
☐	☐	Stress Tests
☐	☐	Non-UI Image Transfer Tests
☐	☐	UI Image Transfer Tests
☐	☐	CAP_XFERCOUNT
☐	☐	Version Tests

I attest under penalty of perjury to the fact that the information on this form is true and accurate.

____________________ ____________________
Signature of Authorized Representative Date

____
Printed Name

Subscribed and duly sworn in my presence this ______ day of ______________ 20___.

Country of ________________ State of _______________________

____
Notary Public Signature

SS

My commission expires: _______________________________

Mail the Notarized Document to:

The TWAIN Working Group
4256 Redspire Lane
Fayetteville, NC 28306 USA

- or -

Email the Digitally Signed Document to:

admin@twain.org

Sample Form

Affirmation of Successful Completion of TWAIN Self-Certification
Compliance with TWAIN Versions 1.9 through 2.2
Page 1 of 2

Completion and submission of a digitally signed or notarized original of this statement to the TWAIN Working Group authorizes the authorized representative or their company to display the TWAIN Certified Logo on the hardware, software and marketing materials of the TWAIN data source described below. All fields must be filled in, except where otherwise indicated.

The certification mark is intended for use by authorized entities or persons and is intended to certify that this software conforms to standards designated by the TWAIN Working Group. This document indicates compliance with the TWAIN Specification for version TWAIN 2.2 or earlier.

The following information will not be published or shared. The Logo will be sent to the email address.

Name of Contact: __John Smith______________________________

Email Address: __twainselfcert@notarealcompany.com__________

May the TWAIN Working Group send TWAIN information not related to this submission to this email address? (circle one) [Yes] [No]

The following fields will be posted on the TWAIN Working Group website.

Company:	Not A Real Company
Division: (optional)	Scanner Group
Company/Division URL: (optional)	www.notarealcompany.com/scanners
Hardware Model Name:	Business Scanner
Hardware Model Number:	123
Hardware Model Revision: (optional)	6.0
TW_IDENTITY.Manufacturer:	Not A Real Company
TW_IDENTITY.ProductFamily:	Business Scanner
TW_IDENTITY.ProductName:	Not A Real Scanner: 123
TW_IDENTITY.Protocol:	__2___. __1___
TWAIN Data Source Version:	___5___. ___3___ . ___0___ . ___0___
Installer Version:	Not A Real Scanner: 123, CD v3.4.0.0
URL to Data Source: (optional)	www.notarealcompany.com/scanners/123
Processor:	x86 _x_ x64 _x_ other ________
Operating System/Revision:	Windows Vista / SP2
Self-Certification Software: (optional)	Inspector TWAIN 3.1.14

May the TWAIN Working Group post the software used to self-certify? (circle one)
[Yes] [No]

Affirmation of Successful Completion of TWAIN Self-Certification
Compliance with TWAIN Versions 1.9 through 2.2
Page 2 of 2

Please confirm that all tests described within the "TWAIN Self-Certification Process for Data Sources" document have been completely and successfully run (check all that apply).

32-bit 64-bit Test

X	X	TWAIN Standard Capability Tests
X	X	Vendor Custom Capability Tests
X	X	Status Return Tests
X	X	Stress Tests
X	X	Non-UI Image Transfer Tests
X	X	UI Image Transfer Tests
X	X	CAP_XFERCOUNT
X	X	Version Tests

I attest under penalty of perjury to the fact that the information on this form is true and accurate.

_______________________________________ _____________________

Signature of Authorized Representative Date

____

Printed Name

Subscribed and duly sworn in my presence this ______ day of _______________ 20___.

Country of _________________ State of ________________________

____

Notary Public Signature

SS

My commission expires: _____________________________

Mail the Notarized Document to:

The TWAIN Working Group
4256 Redspire Lane
Fayetteville, NC 28306

- or -

Email the Digitally Signed Document to:

admin@twain.org

TWAIN "Congratulations" Webpage

Applications that automate the TWAIN self-certification process are asked to use the "Congratulations" web page to complete the process. Hard coding the "Affirmation of Successful Completion of TWAIN Self-Certification" may require updates to the application if the TWAIN Working Group changes the document. Use of the web page avoids this problem.

The URL of the web page is:

http://www.twain.org/self_certification_congratulations.shtm

TWAIN Self-Certification Tests

The tests are broken down into the following groups:

TWAIN Standard Capability Tests	Exercise `DAT_CAPABILITY` operations for all standard TWAIN capabilities reported by `CAP_SUPPORTEDCAPS`. Confirm use of containers and supported operations.
Vendor Custom Capability Tests	Exercise `DAT_CAPABILITY` operations for any vendor specific custom capabilities reported by `CAP_SUPPORTEDCAPS`.
Status Return Tests	Confirm that the expected status return is reported by certain operations.
Stress Tests	Stress aspects of data sources that have been reported as common problems.
Non-UI Image Transfer Tests	Confirm that multiple `MSG_ENABLEDS` and `MSG_DISABLEDS` calls can be made in the context of one `MSG_OPENDS` / `MSG_CLOSEDS`. This test focuses on image capture with no UI.

UI Image Transfer Tests	Confirm that multiple MSG_ENABLEDS and MSG_DISABLEDS calls can be made in the context of one MSG_OPENDS / MSG_CLOSEDS. This test focuses on image capture with the UI.
ICAP_XFERMECH	Test the ability of the data source to transfer the correct number of images based on the value of ICAP_XFERMECH.
Version Test	Confirm that the data sources responds correctly to different TWAIN versions of data source manager and application.

TWAIN Standard Capability Tests

Purpose

Exercise all of the TWAIN Standard capabilities exposed by CAP_SUPPORTEDCAPS using the standard operations supported by DG_CONTROL / DAT_CAPABILITY.

Operations on capabilities (MSG_* values specified below) are assumed to be DG_CONTROL / DAT_CAPABILITY, unless otherwise stated.

Pre-Test Procedure

Open the data source manager. It is required that when opened the data source is in the state it would be in after being installed (e.g., no saved settings from previous sessions), to make the test more reproducible.

When performing this test on Windows Vista or later, Macintosh OS X or Linux it must be successfully completed using both a native 32-bit and a native 64-bit data source.

Confirm Basic Negotiation with CAP_SUPPORTEDCAPS

Make sure that CAP_SUPPORTEDCAPS is working properly. Perform basic checks on how well it supports negotiation.

1. **Action**: MSG_GET CAP_SUPPORTEDCAPS (get the list of capabilities to be tested)

   1.1. **Test**: If result is not TWRC_SUCCESS, then end with error

   1.2. **Test**: If TW_CAPABILITY.Cap is not CAP_SUPPORTEDCAPS, then end with error

   1.3. **Test**: If TW_CAPABILITY.ConType is not TWON_ARRAY, then end with error

   1.4. **Test**: If TW_CAPABILITY.hContainer is not a valid TW_HANDLE value, then end with error

1.5. **Test**: If `TW_ARRAY.ItemType` is not `TWTY_UINT16`, then end with error

1.6. **Test**: If `TW_ARRAY.NumItems` is equal to zero, then end with error

1.7. **Action**: Confirm the presence of the following capabilities in `TW_ARRAY.ItemList`

1.7.1. **Test**: If `CAP_SUPPORTEDCAPS` not found, then end with error

1.7.2. **Test**: If `ICAP_PIXELTYPE` not found, then end with error

1.7.3. **Test**: If `ICAP_XFERMECH` not found, then end with error

Confirm Basic Negotiation with ICAP_PIXELTYPE

Make sure that `ICAP_PIXELTYPE` is working properly.   Perform basic checks on how well it supports negotiation.

2. **Action**: `MSG_GET ICAP_PIXELTYPE`

2.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

2.2. **Test**: If `TW_CAPABILITY.Cap` is not `ICAP_PIXELTYPE`, then end with error

2.3. **Test**: If `TW_CAPABILITY.ConType` is not `TWON_ENUMERATION`, then end with error

2.4. **Test**: If `TW_CAPABILITY.hContainer` is not a valid `TW_HANDLE` value, then end with error

2.5. **Test**: If `TW_ENUMERATION.ItemType` is not `TWTY_UINT16`, then end with error

2.6. **Test**: If `TW_ENUMERATION.NumItems` is equal to zero, then end with error

Confirm Basic Negotiation with ICAP_BITDEPTH

Make sure that `ICAP_BITDEPTH` is working properly, and doesn't include invalid values for commonly used pixel types.Make sure that `ICAP_BITDEPTH` is working properly, and doesn't include invalid values for commonly used pixel types.

3. **Action**: `MSG_SET ICAP_PIXELTYPE` to `TWPT_BW`

3.1. **Test**: If result is not `TWRC_SUCCESS`, then proceed to the `TWPT_GRAY` test immediately below

3.2. **Action**: `MSG_GET ICAP_BITDEPTH`

3.2.1. **Test**: If `TW_CAPABILITY.ConType` is not `TWON_ENUMERATION`, then proceed to the `TWPT_RGB` test below

3.2.2. **Test**: If `TW_CAPABILITY.hContainer` is not a valid `TW_HANDLE` value, then end with error

3.2.3. **Test**: If `TW_ENUMERATION.ItemType` is not `TWTY_UINT16`, then end with error

3.2.4. **Test**: If the `TW_ENUMERATION.ItemList` includes a value of `24`, then end with error

4. **Action**: MSG_SET ICAP_PIXELTYPE to TWPT_GRAY

    4.1. **Test**: If result is not TWRC_SUCCESS, then proceed to the TWPT_RGB test below

    4.2. **Action**: MSG_GET ICAP_BITDEPTH

        4.2.1. **Test**: If TW_CAPABILITY.ConType is not TWON_ENUMERATION, then proceed to the TWPT_RGB test below

        4.2.2. **Test**: If TW_CAPABILITY.hContainer is not a valid TW_HANDLE value, then end with error

        4.2.3. **Test**: If TW_ENUMERATION.ItemType is not TWTY_UINT16, then end with error

        4.2.4. **Test**: If the TW_ENUMERATION.ItemList includes a value of 1, then end with error

        4.2.5. **Test**: If the TW_ENUMERATION.ItemList includes a value of 24, then end with error

5. **Action**: MSG_SET ICAP_PIXELTYPE to TWPT_RGB

    5.1. **Test**: If result is not TWRC_SUCCESS, then proceed to the next test section

    5.2. **Action**: MSG_GET ICAP_BITDEPTH

        5.2.1. **Test**: If TW_CAPABILITY.ConType is not TWON_ENUMERATION, then proceed to the TWPT_RGB test below

        5.2.2. **Test**: If TW_CAPABILITY.hContainer is not a valid TW_HANDLE value, then end with error

        5.2.3. **Test**: If TW_ENUMERATION.ItemType is not TWTY_UINT16, then end with error

        5.2.4. **Test**: If the TW_ENUMERATION.ItemList includes a value of 1, then end with error

Confirm Basic Negotiation with ICAP_XFERMECH

Make sure that ICAP_XFERMECH is working properly. Perform basic checks on how well it supports negotiation.

6. **Action**: MSG_GET ICAP_XFERMECH

    6.1. **Test**: If result is not TWRC_SUCCESS, then end with error

    6.2. **Test**: If TW_CAPABILITY.Cap is not ICAP_XFERMECH, then end with error

    6.3. **Test**: If TW_CAPABILITY.ConType is not TWON_ENUMERATION, then end with error

    6.4. **Test**: If TW_CAPABILITY.hContainer is not a valid TW_HANDLE value, then end with error

    6.5. **Test**: If TW_ENUMERATION.ItemType is not TWTY_UINT16, then end with error

6.6. **Test**: If `TW_ENUMERATION.NumItems` is less than two, then end with error

Exercise DAT_CAPABILITY

Exercise `DAT_CAPABILITY` operations for all TWAIN Standard capabilities (ID's with a value less than 0x8000). Ignore Vendor Custom capabilities (ID's with a value of 0x8000 or greater). Confirm correct ConType and ItemType values described in the TWAIN Specification in the chapter titled Chapter 10, "Capabilities".

7. **Action**: `MSG_RESETALL`

7.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

7.2. **Action**: Repeat this section for each enumerated value found inside of `ICAP_PIXELTYPE`, (testing is done for each value of `ICAP_PIXELTYPE`, to provide the best chance of exercising every available capability)

7.3. **Action**: Repeat this section for Standard TWAIN array values found inside of `CAP_SUPPORTEDCAPS` (each Standard TWAIN capability ID is referred to as #CAP# for the rest of this section)

7.3.1. **Action**: `MSG_QUERYSUPPORT #CAP#`

7.3.1.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

7.3.1.2. **Test**: If `TW_CAPABILITY.Cap` is not `#CAP#`, then end with error

7.3.1.3. **Test**: If `TW_CAPABILITY.ConType` is not `TWON_ONEVALUE`, then end with error

7.3.1.4. **Test**: If `TW_ONEVALUE.ItemType` is not `TWTY_UINT32`, then end with error

7.3.1.5. **Test**: If `TW_CAPABILITY.hContainer` is not a valid `TW_HANDLE` value, then end with error

7.3.1.6. **Test**: If the value of `TW_ONEVALUE.Item` doesn't match the `TWQC` values for this capability, then end with error

7.3.1.7. **Test**: If `TWQC_GET`, `TWQC_GETCURRENT` or `TWQC_GETDEFAULT` is detected, then all three must be present, if any are missing end with error

7.3.1.8. **Test**: If `TWQC_RESET` or `TWQC_SET` is detected, then both must be present, plus `TWQC_GET`, `TWQC_GETCURRENT` and `TWQC_GETDEFAULT`, if not true then end with error

7.3.2. **Action**: If `TWQC_GET` is reported, then call `MSG_GET #CAP#`

7.3.2.1. **Test**: If result is `TWRC_FAILURE / TWCC_CAPSEQERROR`, then skip to the next capability

7.3.2.2. **Test**: If result is not `TWRC_SUCCESS`, then end with error

7.3.2.3. **Test**: If `TW_CAPABILITY.Cap` is not `#CAP#`, then end with error

7.3.2.4. **Test**: If `TW_CAPABILITY.hContainer` is not a valid `TW_HANDLE` value, then end with error

7.3.2.5. **Test**: If the value of `TW_CAPABILITY.ConType` doesn't match the Specification's `MSG_GET` container for this capability, then end with error

7.3.2.6. **Test**: If container's `ItemType` doesn't match the Specification's `ItemType` for this capability, then end with error

7.3.3. **Action**: If `TWQC_GETCURRENT` is reported, then call `MSG_GETCURRENT #CAP#`

7.3.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

7.3.3.2. **Test**: If `TW_CAPABILITY.Cap` is not `#CAP#`, then end with error

7.3.3.3. **Test**: If `TW_CAPABILITY.hContainer` is not a valid `TW_HANDLE` value, then end with error

7.3.3.4. **Action**: If `TWQC_GET` was reported by `MSG_QUERYSUPPORT` then do the following:

7.3.3.4.1. **Test**: If the `TW_CAPABILITY.ConType` for `MSG_GET` was `TWON_ENUMERATION`, `TWON_ONEVALUE` or `TWON_RANGE`, then the `TW_CAPABILITY.ConType` for `MSG_GETCURRENT` must be `TWTY_ONEVALUE`, if not then end with error

7.3.3.4.2. **Test**: If the `TW_CAPABILITY.ConType` for `MSG_GET` was `TWON_ARRAY` then the `TW_CAPABILITY.ConType` for `MSG_GETCURRENT` must be `TWTY_ARRAY`, if not then end with error

7.3.3.4.3. **Test**: If container's ItemType for `MSG_GET` doesn't match container's `ItemType` for `MSG_GETCURRENT`, then end with error

7.3.4. **Action**: If `TWQC_GETDEFAULT` is reported, then call `MSG_GETDEFAULT #CAP#`

7.3.4.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

7.3.4.2. **Test**: If `TW_CAPABILITY.Cap` is not `#CAP#`, then end with error

7.3.4.3. **Test**: If `TW_CAPABILITY.hContainer` is not a valid `TW_HANDLE` value, then end with error

7.3.4.4. **Action**: If `TWQC_GET` was reported by `MSG_QUERYSUPPORT` then do the following:

7.3.4.4.1. **Test**: If the `TW_CAPABILITY.ConType` for `MSG_GET` was `TWON_ENUMERATION`, `TWON_ONEVALUE` or `TWON_RANGE`, then the `TW_CAPABILITY.ConType` for `MSG_GETDEFAULT` must be `TWTY_ONEVALUE`, if not then end with error

7.3.4.4.2. **Test**: If the `TW_CAPABILITY.ConType` for `MSG_GET` was `TWON_ARRAY` then the `TW_CAPABILITY.ConType` for `MSG_GETDEFAULT` must be `TWTY_ARRAY`, if not then end with error

7.3.4.4.3. **Test**: If container's ItemType for `MSG_GET` doesn't match container's `ItemType` for `MSG_GETDEFAULT`, then end with error

7.3.5. **Action**: If `TWQC_RESET` is reported, then call `MSG_RESET #CAP#`

7.3.5.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

7.3.5.2. **Test**: If `TW_CAPABILITY.Cap` is not `#CAP#`, then end with error

7.3.5.3. **Test**: If `TW_CAPABILITY.hContainer` is not a valid `TW_HANDLE` value, then end with error

7.3.6. **Action**: If `TWQC_GET` was reported by `MSG_QUERYSUPPORT` then do the following:

7.3.6.1. **Test**: If T`W_CAPABILITY.ConType` for `MSG_GET` doesn't match `TW_CAPABILITY.ConType` for `MSG_RESET`, then end with error

7.3.6.2. **Test**: If container's `ItemType` for `MSG_GET` doesn't match container's `ItemType` for `MSG_RESET`, then end with error

7.3.7. **Action**: If `TWQC_SET` is reported then do the following:

7.3.7.1. **Action**: If `TWQC_GET` was reported by `MSG_QUERYSUPPORT` then do the following:

7.3.7.1.1. **Action**: `MSG_GET #CAP#`

7.3.7.1.1.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

7.3.7.1.2. **Action**: `MSG_SET` with `TW_CAPABILITY` from `MSG_GET`

7.3.7.1.2.1. **Test**: If result is `TWRC_FAILURE` / `TWCC_CAPSEQERROR`, then skip to next capability

7.3.7.1.2.2. **Test**: If result is not `TWRC_SUCCESS` or `TWRC_CHECKSTATUS`, then end with error

7.3.7.2. **Action**: If `TWQC_GETCURRENT` was reported by `MSG_QUERYSUPPORT` then do the following:

7.3.7.2.1. **Action**: `MSG_GETCURRENT #CAP#`

7.3.7.2.1.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

7.3.7.2.1.2. **Action**: `MSG_SET` with `TW_CAPABILITY` from `MSG_GETCURRENT`

7.3.7.2.1.3. **Test**: If result is `TWRC_FAILURE` / `TWCC_CAPSEQERROR`, then skip to next capability

7.3.7.2.1.4. **Test**: If result is not `TWRC_SUCCESS` or `TWRC_CHECKSTATUS`, then end with error

7.3.7.3. **Action**: If `TWQC_GETDEFAULT` was reported by `MSG_QUERYSUPPORT` then do the following:

7.3.7.3.1. **Action**: `MSG_GETDEFAULT #CAP#`

7.3.7.3.1.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

7.3.7.3.2. **Action**: `MSG_SET` with `TW_CAPABILITY` from `MSG_GETDEFAULT`

7.3.7.3.2.1. **Test**: If result is `TWRC_FAILURE` / `TWCC_CAPSEQERROR`, then skip to next capability

7.3.7.3.2.2. **Test**: If result is not `TWRC_SUCCESS` or `TWRC_CHECKSTATUS`, then end with error

7.3.7.4. **Action**: If `TWQC_RESET` was reported by `MSG_QUERYSUPPORT` then do the following:

7.3.7.4.1. **Action**: `MSG_RESET #CAP#`

7.3.7.4.1.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

7.3.7.4.2. **Action**: `MSG_SET` with `TW_CAPABILITY` from `MSG_RESET`

7.3.7.4.2.1. **Test**: If result is `TWRC_FAILURE` / `TWCC_CAPSEQERROR`, then skip to next capability

7.3.7.4.2.2. **Test**: If result is not TWRC_SUCCESS, then end with error

7.3.7.5. **Action**: If TWQC_GET was reported by MSG_QUERYSUPPORT then do the following:

7.3.7.5.1. **Action**: MSG_GET #CAP#

7.3.7.5.1.1. **Test**: If result is not TWRC_SUCCESS, then end with error

7.3.7.5.1.2. **Test**: If the container's ItemType is TWTY_BOOL and the test application has DF_APP2 in its TW_IDENTITY.SupportedGroups, and the data source has DF_DS2 in its TW_IDENTITY.SupportedGroups, then TW_CAPABILITY.ConType must be set to TW_ENUMERATION, if not then end with error

7.3.7.5.1.3. **Test**: If the container's ItemType is TWTY_BOOL and the test application does not have DF_APP2 in its TW_IDENTITY.SupportedGroups, or the data source does not have DF_DS2 in its TW_IDENTITY.SupportedGroups, then TW_CAPABILITY.ConType must be set to TW_ONEVALUE, if not then end with error

7.3.7.5.2. **Action**: If TW_CAPABILITY.ConType is TWON_ARRAY then repeat following for each value in the array:

7.3.7.5.2.1. **Action**: MSG_SET the value using a TW_ARRAY container

7.3.7.5.2.1.1. **Test**: If result is not TWRC_SUCCESS or TWRC_CHECKSTATUS, then end with error

7.3.7.5.2.2. **Action**: If TW_CAPABILITY.ConType is TWON_ARRAY then do the following:

7.3.7.5.2.2.1. **Action**: MSG_SET the value using a TW_ARRAY container, setting the value to 22222 (which is expected to be an illegal value)

7.3.7.5.2.3. **Test**: If result is not TWRC_BADVALUE or TWRC_CHECKSTATUS, then end with error

7.3.7.5.3. **Action**: If TW_CAPABILITY.ConType is TWON_ENUMERATION then repeat following for each value in the enumeration:

7.3.7.5.4. **Action**: MSG_SET the value using a TW_ENUMERATION container

7.3.7.5.4.1. **Test**: If result is not TWRC_SUCCESS or TWRC_CHECKSTATUS, then end with error

7.3.7.5.5. **Action**: If TW_CAPABILITY.ConType is TWON_ENUMERATION then do the following:

7.3.7.5.5.1. **Action**: MSG_SET the current value using a TW_ONEVALUE container, the value must be something that did not appear in the list of valid enumerations

7.3.7.5.5.1.1. **Test**: If result is not TWRC_BADVALUE, then end with error

7.3.7.5.6. **Action**: If TW_CAPABILITY.ConType is TWON_RANGE then repeat the following for the TW_RANGE.MinValue, TW_RANGE.CurrentValue and TW_RANGE.MaxValue:

7.3.7.5.6.1. **Action**: MSG_SET the current value using a TW_RANGE container

7.3.7.5.6.1.1. **Test**: If result is not TWRC_SUCCESS or TWRC_CHECKSTATUS, then end with error

Post-Test Procedure

When testing is completed, close the data source and the data source manager.

Vendor Custom Capability Tests

Purpose

Exercise all of the Vendor Custom capabilities exposed by CAP_SUPPORTEDCAPS using the standard operations supported by DG_CONTROL / DAT_CAPABILITY.

Operations on capabilities (MSG_* values specified below) are assumed to be DG_CONTROL / DAT_CAPABILITY, unless otherwise stated.

Pre-Test Procedure

Open the data source manager and the data source that is to be tested. It is recommended that the data source is in the state it would be in after being installed (e.g., no saved settings from previous sessions), to make the test more reproducible.

When performing this test on Windows Vista or later, Macintosh OS X or Linux it must be successfully completed using both a native 32-bit and a native 64-bit data source.

Confirm Basic Negotiation with CAP_SUPPORTEDCAPS

Make sure that CAP_SUPPORTEDCAPS is working properly. Perform basic checks on how well it supports negotiation.

1. **Action**: MSG_RESETALL

   1.1. **Test**: If return code is not TWRC_SUCCESS, end with an error

   1.2. **Action**: MSG_GET CAP_SUPPORTEDCAPS (gets the list of capabilities to be tested)

      1.2.1. **Test**: If result is not TWRC_SUCCESS, then end with error

      1.2.2. **Test**: If TW_CAPABILITY.Cap is not CAP_SUPPORTEDCAPS, then end with error

      1.2.3. **Test**: If TW_CAPABILITY.ConType is not TWON_ARRAY, then end with error

      1.2.4. **Test**: If TW_CAPABILITY.hContainer is not a valid TW_HANDLE value, then end with error

      1.2.5. **Test**: If TW_ARRAY.ItemType is not TWTY_UINT16, then end with error

      1.2.6. **Test**: If TW_ARRAY.NumItems is equal to zero, then end with error

      1.2.7. **Action**: Confirm the presence of the following capabilities in TW_ARRAY.ItemList

         1.2.7.1. **Test**: If CAP_SUPPORTEDCAPS not found, then end with error

         1.2.7.2. **Test**: If ICAP_PIXELTYPE not found, then end with error

Confirm Basic Negotiation with ICAP_PIXELTYPE

Make sure that ICAP_PIXELTYPE is working properly. Perform basic checks on how well it supports negotiation.

2. **Action**: MSG_GET ICAP_PIXELTYPE

   2.1. **Test**: If result is not TWRC_SUCCESS, then end with error

   2.2. **Test**: If TW_CAPABILITY.Cap is not ICAP_PIXELTYPE, then end with error

2.3. **Test**: If `TW_CAPABILITY.ConType` is not `TWON_ENUMERATION`, then end with error

2.4. **Test**: If `TW_CAPABILITY.hContainer` is not a valid `TW_HANDLE` value, then end with error

2.5. **Test**: If `TW_ENUMERATION.ItemType` is not `TWTY_UINT16`, then end with error

2.6. **Test**: If `TW_ENUMERATION.NumItems` is equal to zero, then end with error

Exercise DAT_CAPABILITY

Exercise `DAT_CAPABILITY` operations for all Vendor Custom capabilities (ID's with a value of `0x8000` or greater). Ignore TWAIN Standard capabilities (ID's with a value less than `0x8000`).

3. **Action**: Repeat this section for each enumerated value found inside of `ICAP_PIXELTYPE`, (testing is done for each value of `ICAP_PIXELTYPE`, to provide the best chance of exercising every available capability)

3.1. **Action**: Repeat this section for each Vendor Custom TWAIN array value found inside of `CAP_SUPPORTEDCAPS` (each Vendor Custom capability ID is referred to as `#CAP#` for the rest of this section)

3.1.1. **Action**: `MSG_QUERYSUPPORT #CAP#`

3.1.1.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

3.1.1.2. **Test**: If `TW_CAPABILITY.Cap` is not `#CAP#`, then end with error

3.1.1.3. **Test**: If `TW_CAPABILITY.ConType` is not `TWON_ONEVALUE`, then end with error

3.1.1.4. **Test**: If `TW_ONEVALUE.ItemType` is not `TWTY_UINT32`, then end with error

3.1.1.5. **Test**: If `TW_CAPABILITY.hContainer` is not a valid `TW_HANDLE` value, then end with error

3.1.2. **Action**: If `TWQC_GET` is reported, then call `MSG_GET #CAP#`

3.1.2.1. **Test**: If result is `TWRC_FAILURE / TWCC_CAPSEQERROR`, then skip to the next capability

3.1.2.2. **Test**: If result is not `TWRC_SUCCESS`, then end with error

3.1.2.3. **Test**: If `TW_CAPABILITY.Cap` is not `#CAP#`, then end with error

3.1.2.4. **Test**: If `TW_CAPABILITY.hContainer` is not a valid `TW_HANDLE` value, then end with error

3.1.3. **Action**: If `TWQC_GETCURRENT` is reported, then call `MSG_GETCURRENT #CAP#`

3.1.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

3.1.3.2. **Test**: If `TW_CAPABILITY.Cap` is not `#CAP#`, then end with error

3.1.3.3. **Test**: If `TW_CAPABILITY.hContainer` is not a valid `TW_HANDLE` value, then end with error

3.1.3.4. **Action**: If `TWQC_GET` was reported by `MSG_QUERYSUPPORT` then do the following:

3.1.3.4.1. **Test**: If the `TW_CAPABILITY.ConType` for `MSG_GET` was `TWON_ENUMERATION`, `TWON_ONEVALUE` or `TWON_RANGE`, then the `TW_CAPABILITY.ConType` for `MSG_GETCURRENT` must be `TWTY_ONEVALUE`, if not then end with error

3.1.3.4.2. **Test**: If the `TW_CAPABILITY.ConType` for `MSG_GET` was `TWON_ARRAY` then the `TW_CAPABILITY.ConType` for `MSG_GETCURRENT` must be `TWTY_ARRAY`, if not then end with error

3.1.3.4.3. **Test**: If container's ItemType for `MSG_GET` doesn't match container's `ItemType` for `MSG_GETCURRENT`, then end with error

3.1.4. **Action**: If `TWQC_GETDEFAULT` is reported, then call `MSG_GETDEFAULT #CAP#`

3.1.4.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

3.1.4.2. **Test**: If `TW_CAPABILITY.Cap` is not `#CAP#`, then end with error

3.1.4.3. **Test**: If `TW_CAPABILITY.hContainer` is not a valid `TW_HANDLE` value, then end with error

3.1.4.4. **Action**: If `TWQC_GET` was reported by `MSG_QUERYSUPPORT` then do the following:

3.1.4.4.1. **Test**: If the `TW_CAPABILITY.ConType` for `MSG_GET` was `TWON_ENUMERATION`, `TWON_ONEVALUE` or `TWON_RANGE`, then the `TW_CAPABILITY.ConType` for `MSG_GETDEFAULT` must be `TWTY_ONEVALUE`, if not then end with error

3.1.4.4.2. **Test**: If the `TW_CAPABILITY.ConType` for `MSG_GET` was `TWON_ARRAY` then the `TW_CAPABILITY.ConType` for `MSG_GETDEFAULT` must be `TWTY_ARRAY`, if not then end with error

3.1.4.4.3. **Test**: If container's `ItemType` for `MSG_GET` doesn't match container's `ItemType` for `MSG_GETDEFAULT`, then end with error

3.1.5. **Action**: If `TWQC_RESET` is reported, then call `MSG_RESET #CAP#`

3.1.5.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

3.1.5.2. **Test**: If `TW_CAPABILITY.Cap` is not `#CAP#`, then end with error

3.1.5.3. **Test**: If `TW_CAPABILITY.hContainer` is not a valid `TW_HANDLE` value, then end with error

3.1.5.4. **Action**: If `TWQC_GET` was reported by `MSG_QUERYSUPPORT` then do the following:

3.1.5.4.1. **Test**: If `TW_CAPABILITY.ConType` for `MSG_GET` doesn't match `TW_CAPABILITY.ConType` for `MSG_RESET`, then end with error

3.1.5.4.2. **Test**: If container's `ItemType` for `MSG_GET` doesn't match container's `ItemType` for `MSG_RESET`, then end with error

3.1.6. **Action**: If `TWQC_SET` is reported then do the following:

3.1.6.1. **Action**: If `TWQC_GET` was reported by `MSG_QUERYSUPPORT` then do the following:

3.1.6.1.1. **Action**: `MSG_GET #CAP#`

3.1.6.1.1.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

3.1.6.1.2. **Action**: `MSG_SET` with `TW_CAPABILITY` from `MSG_GET`

3.1.6.1.2.1. **Test**: If result is `TWRC_FAILURE / TWCC_CAPSEQERROR`, then skip to next capability

3.1.6.1.2.2. **Test**: If result is not `TWRC_SUCCESS` or `TWRC_CHECKSTATUS`, then end with error

3.1.6.2. **Action**: If `TWQC_GETCURRENT` was reported by `MSG_QUERYSUPPORT` then do the following:

3.1.6.2.1. **Action**: `MSG_GETCURRENT #CAP#`

3.1.6.2.1.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

3.1.6.2.2. **Action**: `MSG_SET` with `TW_CAPABILITY` from `MSG_GETCURRENT`

3.1.6.2.2.1. **Test**: If result is `TWRC_FAILURE` / `TWCC_CAPSEQERROR`, then skip to next capability

3.1.6.2.2.2. **Test**: If result is not `TWRC_SUCCESS` or `TWRC_CHECKSTATUS`, then end with error

3.1.6.3. **Action**: If `TWQC_GETDEFAULT` was reported by `MSG_QUERYSUPPORT` then do the following:

3.1.6.3.1. **Action**: `MSG_GETDEFAULT #CAP#`

3.1.6.3.1.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

3.1.6.3.2. **Action**: `MSG_SET` with `TW_CAPABILITY` from `MSG_GETDEFAULT`

3.1.6.3.2.1. **Test**: If result is `TWRC_FAILURE / TWCC_CAPSEQERROR`, then skip to next capability

3.1.6.3.2.2. **Test**: If result is not `TWRC_SUCCESS` or `TWRC_CHECKSTATUS`, then end with error

3.1.6.4. **Action**: If `TWQC_RESET` was reported by `MSG_QUERYSUPPORT` then do the following:

3.1.6.4.1. **Action**: `MSG_RESET #CAP#`

3.1.6.4.1.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

3.1.6.4.2. **Action**: `MSG_SET` with `TW_CAPABILITY` from `MSG_RESET`

3.1.6.4.2.1. **Test**: If result is `TWRC_FAILURE / TWCC_CAPSEQERROR`, then skip to next capability

3.1.6.4.2.2. **Test**: If result is not `TWRC_SUCCESS`, then end with error

3.1.6.5. **Action**: If `TWQC_GET` was reported by `MSG_QUERYSUPPORT` then do the following:

3.1.6.5.1. **Action**: `MSG_GET #CAP#`

3.1.6.5.1.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

3.1.6.5.1.2. **Test**: If the container's `ItemType` is `TWTY_BOOL` and the test application has `DF_APP2` in its `TW_IDENTITY.SupportedGroups`, and the data source has `DF_DS2` in its

TW_IDENTITY.SupportedGroups, then TW_CAPABILITY.ConType must be set to TW_ENUMERATION, if not then end with error

3.1.6.5.1.3. **Test**: If the container's ItemType is TWTY_BOOL and the test application does not have DF_APP2 in its TW_IDENTITY.SupportedGroups, or the data source does not have DF_DS2 in its TW_IDENTITY.SupportedGroups, then TW_CAPABILITY.ConType must be set to TW_ONEVALUE, if not then end with error

3.1.6.5.2. **Action**: If TW_CAPABILITY.ConType is TWON_ARRAY then repeat following for each value in the array:

3.1.6.5.2.1. **Action**: MSG_SET the value using a TW_ARRAY container

3.1.6.5.2.1.1. **Test**: If result is not TWRC_SUCCESS or TWRC_CHECKSTATUS, then end with error

3.1.6.5.3. **Action**: If TW_CAPABILITY.ConType is TWON_ARRAY then do the following:

3.1.6.5.3.1. **Action**: MSG_SET the value using a TW_ARRAY container, setting the value to 22222 (which is expected to be an illegal value)

3.1.6.5.3.1.1. **Test**: If result is not TWRC_BADVALUE or TWRC_CHECKSTATUS, then end with error

3.1.6.5.4. **Action**: If TW_CAPABILITY.ConType is TWON_ENUMERATION then repeat following for each value in the enumeration:

3.1.6.5.5. **Action**: MSG_SET the value using a TW_ENUMERATION container

3.1.6.5.5.1. **Test**: If result is not TWRC_SUCCESS or TWRC_CHECKSTATUS, then end with error

3.1.6.5.6. **Action**: If TW_CAPABILITY.ConType is TWON_ENUMERATION then do the following:

3.1.6.5.6.1. **Action**: `MSG_SET` the current value using a `TW_ONEVALUE` container, the value must be something that did not appear in the list of valid enumerations

3.1.6.5.6.1.1. **Test**: If result is not `TWRC_BADVALUE`, then end with error

3.1.6.5.7. **Action**: If `TW_CAPABILITY.ConType` is `TWON_RANGE` then repeat the following for the `TW_RANGE.MinValue`, `TW_RANGE.CurrentValue` and `TW_RANGE.MaxValue`:

3.1.6.5.7.1. **Action**: `MSG_SET` the current value using a `TW_RANGE` container

3.1.6.5.7.1.1. **Test**: If result is not `TWRC_SUCCESS` or `TWRC_CHECKSTATUS`, then end with error

Post-Test Procedure

When testing is completed, close the data source and the data source manager.

Status Return Tests

Purpose

Confirm that the expected status return is reported by certain operations.

This is not an exhaustive test of all possible Status Returns.

Pre-Test Procedure

Open the data source manager and the data source that is to be tested. It is recommended that the data source is in the state it would be in after being installed (e.g., no saved settings from previous sessions), to make the test more reproducible.

When performing this test on Windows Vista or later, Macintosh OS X or Linux it must be successfully completed using both a native 32-bit and a native 64-bit data source.

Check Status Returns for DAT_IMAGENATIVEXFER and DAT_IMAGEMEMXFER

Confirm that `DAT_IMAGENATIVEXFER` and `DAT_IMAGEMEMXFER` both return the correct status returns in various error conditions.

1. **Action**: In State 4 (after `MSG_OPENDS`, but before calling `MSG_ENABLEDS`)…

1.1. Confirm that the proper statuses are returned for bad protocols and attempts to perform image transfers in State 4.

1.2. **Action**: Call `DG_IMAGE / DAT_IMAGENATIVEXFER / MSG_SET`

1.2.1. **Test**: If result is not `TWRC_FAILURE / TWCC_BADPROTOCOL`, then end with error

1.3. **Action**: Call `DG_IMAGE / DAT_IMAGENATIVEXFER / MSG_GET`

1.3.1. **Test**: If result is not `TWRC_FAILURE / TWCC_SEQERROR`, then end with error

1.4. **Action**: Call `DG_IMAGE / DAT_IMAGEMEMXFER / MSG_SET`

1.4.1. **Test**: If result is not `TWRC_FAILURE / TWCC_BADPROTOCOL`, then end with error

1.5. **Action**: Call `DG_IMAGE / DAT_IMAGEMEMXFER / MSG_GET`

1.5.1. **Test**: If result is not `TWRC_FAILURE / TWCC_SEQERROR`, then end with error

Check Status Returns for DAT_IMAGELAYOUT

Confirm that `DAT_IMAGELAYOUT` returns the correct status returns in various error conditions.

2. **Action**: Call `DG_CONTROL / DAT_USERINTERFACE / MSG_ENABLEDS` with `ShowUI = TRUE`

2.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

2.2. **Action**: Call `DG_IMAGE / DAT_IMAGELAYOUT / MSG_GET`

2.2.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

2.3. **Action**: Call `DG_IMAGE / DAT_IMAGELAYOUT / MSG_SET` using the `TW_IMAGELAYOUT` values from the previous `MSG_GET` call

2.3.1. **Test**: If result is not `TWRC_FAILURE / TWRC_SEQERROR`, then end with error

2.4. **Action**: Call `DG_IMAGE / DAT_IMAGELAYOUT / MSG_RESET`

2.4.1. **Test**: If result is not `TWRC_FAILURE / TWCC_SEQERROR`, then end with error

Check Status Returns for DAT_CAPABILITY

Confirm that `DAT_CAPABILITY` returns the correct status returns in various error conditions.

3. **Action**: Call `DG_CONTROL / DAT_USERINTERFACE / MSG_ENABLEDS` with `ShowUI = TRUE`

3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

3.2. **Action**: `MSG_GET CAP_SUPPORTEDCAPS`

  3.2.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

3.3. **Action**: `MSG_GET CAP_EXTENDEDCAPS`

  3.3.1. **Test**: If result is not `TWRC_SUCCESS` or the `TW_ARRAY` is empty, then skip any checks of `CAP_EXTENDEDCAPS` referenced in the rest of this section

3.4. **Action**: For each value found in `CAP_SUPPORTEDCAPS` that is not in `CAP_EXTENDEDCAPS` do the following sections (each capability ID is referred to as `#CAP#` for the rest of this section):

  3.4.1. **Action**: `MSG_GET #CAP#`

    3.4.1.1. **Test**: If result is not `TWRC_SUCCESS`, then skip to next capability

  3.4.2. **Action**: `MSG_SET #CAP#` with results of previous `MSG_GET`

    3.4.2.1. **Test**: If result is `TWRC_SUCCESS` or `TWRC_CHECKSTATUS`, then end with error

  3.4.3. **Action**: `MSG_RESET #CAP#`

    3.4.3.1. **Test**: If result is `TWRC_SUCCESS` or `TWRC_CHECKSTATUS`, then end with error

Post-Test Procedure

When testing is completed, close the data source and the data source manager.

Stress Tests

Purpose

Stress aspects of data sources that have been reported as common problems.

Pre-Test Procedure

Open the data source manager. It is required that when opened the data source is in the state it would be in after being installed (e.g., no saved settings from previous sessions), to make the test more reproducible.

When performing this test on Windows Vista or later, Macintosh OS X or Linux it must be successfully completed using both a native 32-bit and a native 64-bit data source.

Open and Close the Data Source Multiple Times

Confirm that the data source can open and close multiple times. This tests for crashes.

1. **Action**: Repeat this section twenty (20) times

1.1. Confirm that the data source can successfully open and close repeated times from a single instance of an application.

1.2. **Action**: Call `DG_CONTROL / DAT_IDENTITY / MSG_OPENDS`

1.2.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.3. **Action**: Call `DG_CONTROL / DAT_IDENTITY / MSG_CLOSEDS`

1.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

Post-Test Procedure

When testing is completed, close the data source and the data source manager.

Non-UI Image Transfer Tests

Purpose

Confirm that multiple `MSG_ENABLEDS` and `MSG_DISABLEDS` calls can be made in the context of one `MSG_OPENDS / MSG_CLOSEDS`. This test focuses on image capture with no UI, verifying that the Application does not have to close the driver after capturing images.

Pre-Test Procedure

Open the data source manager and the data source that is to be tested. It is recommended that the data source is in the state it would be in after being installed (e.g., no saved settings from previous sessions), to make the test more reproducible.

When performing this test on Windows Vista or later, Macintosh OS X or Linux it must be successfully completed using both a native 32-bit and a native 64-bit data source.

Exercise DAT_IMAGENATIVEXFER

This test issues multiple image transfer sessions using `DAT_IMAGENATIVEXFER`. It is performed for all available image sources (unspecified, flatbed and/or ADF). Only one image is transferred per session.

1. **Action**: `MSG_RESETALL`

1.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

1.2. **Action**: `MSG_GET CAP_SUPPORTEDCAPS` (get the list of capabilities to be tested)

1.3. **Action**: `MSG_SET ICAP_XFERMECH` to `TWSX_NATIVE`

1.4. **Action**: `MSG_GETCURRENT ICAP_XFERMECH`

1.5. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

1.6. **Test**: If value is not `TWSX_NATIVE`, end with an error.

1.7. **Action**: If `CAP_FEEDERENABLED` is `TRUE`, set `CAP_AUTOFEED` to `TRUE`

1.8. **Action**: `MSG_SET CAP_DUPLEXENABLED` to `FALSE`

1.9. **Action**: `MSG_SET CAP_XFERCOUNT` to `1`

1.10. **Action**: Do the following for each supported `ICAP_PIXELTYPE`

- 1.10.1. **Action**: `MSG_SET ICAP_PIXELTYPE`
- 1.10.2. **Action**: `MSG_GET ICAP_BITDEPTH`
- 1.10.3. **Action**: Do the following for each supported `ICAP_BITDEPTH`
  - 1.10.3.1. **Action**: `MSG_SET ICAP_BITDEPTH`
  - 1.10.3.2. **Action**: Do the following for the minimum, maximum and 300 (or nearest) resolution values.
    - 1.10.3.2.1. **Action**: `MSG_SET ICAP_XRESOLUTION` and `ICAP_YRESOLUTION`
    - 1.10.3.2.2. **Action**: `DG_CONTROL / DAT_USERINTERFACE / MSG_ENABLEDS` with `ShowUI = FALSE` and `ModalUI = FALSE`
    - 1.10.3.2.3. **Test**: If return code is not `TWRC_SUCCESS`, end with an error
    - 1.10.3.2.4. **Action**: Wait for `MSG_XFERREADY`
    - 1.10.3.2.5. **Action**: `MSG_GET ICAP_XFERMECH`
    - 1.10.3.2.6. **Test**: If return code is not `TWRC_SUCCESS`, end with an error
    - 1.10.3.2.7. **Action**: `DG_IMAGE / DAT_IMAGENATIVEXFER / MSG_GET`
    - 1.10.3.2.8. **Test**: If return code is not `TWRC_XFERDONE`, end with an error
    - 1.10.3.2.9. **Test**: If the handle does not point to a valid image, end with an error
    - 1.10.3.2.10. **Test**: If the bit depth of the image is not what was requested, end with an error
    - 1.10.3.2.11. **Action**: Free handle returned by `DAT_IMAGENATIVEXFER`
    - 1.10.3.2.12. **Action**: `DG_CONTROL / DAT_PENDINGXFERS / MSG_ENDXFER`

1.10.3.2.13. **Action**: DG_CONTROL / DAT_USERINTERFACE / MSG_DISABLEDS

1.10.3.2.14. **Test**: If return code is not TWRC_SUCCESS, end with an error

Exercise DAT_IMAGEMEMXFER

This test issues multiple image transfer sessions using DAT_IMAGEMEMXFER. It is performed for all available image sources (unspecified, flatbed and/or ADF). Only one image is transferred per session. The preferred size specified by the data source is used to transfer each strip.

2. **Action**: MSG_RESETALL

2.1. **Test**: If return code is not TWRC_SUCCESS, end with an error

2.2. **Action**: MSG_SET ICAP_XFERMECH to TWSX_MEMORY

2.3. **Action**: MSG_GETCURRENT ICAP_XFERMECH

2.4. **Test**: If return code is not TWRC_SUCCESS, end with an error

2.5. **Test**: If value is not TWSX_MEMORY, end with an error

2.6. **Action**: If CAP_FEEDERENABLED is TRUE, set CAP_AUTOFEED to TRUE

2.7. **Action**: MSG_SET CAP_DUPLEXENABLED to FALSE

2.8. **Action**: MSG_SET CAP_XFERCOUNT to 1

2.9. **Action**: Do the following for each supported ICAP_PIXELTYPE

2.9.1. **Action**: MSG_SET ICAP_PIXELTYPE

2.9.2. **Action**: MSG_GET ICAP_BITDEPTH

2.9.3. **Action**: Do the following for each supported ICAP_BITDEPTH

2.9.3.1. **Action**: MSG_SET ICAP_BITDEPTH

2.9.3.2. **Action**: MSG_GET ICAP_COMPRESSION

2.9.3.3. **Action**: Do the following for each supported ICAP_COMPRESSION

2.9.3.3.1. **Action**: MSG_SET ICAP_COMPRESSION

2.9.3.3.2. **Action**: Do the following for the minimum, maximum and 300 (or nearest) resolution values.

2.9.3.3.2.1. **Action**: MSG_SET ICAP_XRESOLUTION and ICAP_YRESOLUTION

2.9.3.3.2.2. **Action**: `DG_CONTROL / DAT_USERINTERFACE / MSG_ENABLEDS` with `ShowUI = FALSE` and `ModalUI = FALSE`

2.9.3.3.2.3. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

2.9.3.3.2.4. **Action**: Wait for `MSG_XFERREADY`

2.9.3.3.2.5. **Action**: `MSG_GET ICAP_XFERMECH`

2.9.3.3.2.6. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

2.9.3.3.2.7. **Action**: `DG_CONTROL / DAT_SETUPMEMXFER / MSG_GET`

2.9.3.3.2.8. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

2.9.3.3.2.9. **Action**: `DG_IMAGE / DAT_IMAGEMEMXFER / MSG_GET` with the preferred buffer size

2.9.3.3.2.10. **Test**: if the return code is `TWRC_SUCCESS`, repeat previous step

2.9.3.3.2.11. **Test**: if the return code is not `TWRC_XFERDONE`, end with an error

2.9.3.3.2.12. **Action**: `DG_CONTROL / DAT_PENDINGXFERS / MSG_ENDXFER`

2.9.3.3.2.13. **Action**: `DG_CONTROL / DAT_USERINTERFACE / MSG_DISABLEDS`

2.9.3.3.2.14. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

Exercise DAT_IMAGEFILEXFER

This test issues multiple image transfer sessions using `DAT_IMAGEFILEXFER`. It is performed for all available image sources (unspecified, flatbed and/or ADF). Only one image is transferred per session. The preferred size specified by the data source is used to transfer each strip.

3. **Action**: `MSG_RESETALL`

   3.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

- 3.2. **Action**: `MSG_SET ICAP_XFERMECH` to `TWSX_MEMORY`
- 3.3. **Test**: If return code is `TWRC_SUCCESS / TWCC_BADVALUE`, skip to section 4
- 3.4. **Test**: If return code is not `TWRC_SUCCESS`, end with an error
- 3.5. **Action**: `MSG_SET ICAP_XFERMECH` to `TWSX_FILE`
- 3.6. **Action**: If `CAP_FEEDERENABLED` is `TRUE`, set `CAP_AUTOFEED` to `TRUE`
- 3.7. **Action**: `MSG_SET CAP_DUPLEXENABLED` to `FALSE`
- 3.8. **Action**: `MSG_SET CAP_XFERCOUNT` to `1`
- 3.9. **Action**: `MSG_GET ICAP_IMAGEFILEFORMAT`
- 3.10. **Action**: Do the following for each supported `ICAP_IMAGEFILEFORMAT`
  - 3.10.1. **Action**: `MSG_SET ICAP_IMAGEFILEFORMAT`
  - 3.10.2. **Action**: `MSG_GET ICAP_PIXELTYPE`
  - 3.10.3. **Action**: Do the following for each supported `ICAP_PIXELTYPE`
    - 3.10.3.1. **Action**: `MSG_SET ICAP_PIXELTYPE`
    - 3.10.3.2. **Action**: `MSG_GET ICAP_BITDEPTH`
    - 3.10.3.3. **Action**: Do the following for each supported `ICAP_BITDEPTH`
      - 3.10.3.3.1. **Action**: `MSG_SET ICAP_BITDEPTH`
      - 3.10.3.3.2. **Action**: `MSG_GET ICAP_COMPRESSION`
      - 3.10.3.3.3. **Action**: Do the following for each supported `ICAP_COMPRESSION`
        - 3.10.3.3.3.1. **Action**: `MSG_SET ICAP_COMPRESSION`
        - 3.10.3.3.3.2. **Action**: Do the following for the minimum, maximum and 300 (or nearest) resolution values.
          - 3.10.3.3.3.2.1. **Action**: `MSG_SET ICAP_XRESOLUTION` and `ICAP_YRESOLUTION`
          - 3.10.3.3.3.2.2. **Action**: `DG_CONTROL / DAT_USERINTERFACE / MSG_ENABLEDS` with `ShowUI = FALSE` and `ModalUI = FALSE`

3.10.3.3.3.2.3. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

3.10.3.3.3.2.4. **Action**: Wait for `MSG_XFERREADY`

3.10.3.3.3.2.5. **Action**: `MSG_GET ICAP_XFERMECH`

3.10.3.3.3.2.6. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

3.10.3.3.3.2.7. **Action**: `DG_CONTROL / DAT_SETUPFILEXFER / MSG_SET`

3.10.3.3.3.2.8. **Action**: `DG_IMAGE / DAT_IMAGEFILEXFER / MSG_GET`

3.10.3.3.3.2.9. **Test**: If return code is not `TWRC_XFERDONE`, end with an error

3.10.3.3.3.2.10. **Action**: `DG_CONTROL / DAT_PENDINGXFERS / MSG_ENDXFER`

3.10.3.3.3.2.11. **Action**: `DG_CONTROL / DAT_USERINTERFACE / MSG_DISABLEDS`

3.10.3.3.3.2.12. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

Post-Test Procedure

When testing is completed, close the data source and the data source manager.

UI Image Transfer Tests

Purpose

Confirm that multiple `MSG_ENABLEDS` and `MSG_DISABLEDS` calls can be made in the context of one `MSG_OPENDS / MSG_CLOSEDS`. This test focuses on image capture with the UI, verifying that the Application does not have to close the driver after capturing images.

Procedure

These tests are identical to the "Non-UI Image Transfer Tests", except that the value of `ShowUI` is set to `TRUE` instead of `FALSE`.

When performing this test on Windows Vista or later, Macintosh OS X or Linux it must be successfully completed using both a native 32-bit and a native 64-bit data source.

CAP_XFERCOUNT Tests

Purpose

Confirm that when the data source accepts various values for `CAP_XFERCOUNT`, that it returns the specified number of images. Test both flatbed and document feeders.

Pre-Test Procedure

Open the data source manager and the data source that is to be tested. It is recommended that the data source is in the state it would be in after being installed (e.g., no saved settings from previous sessions), to make the test more reproducible.

When performing this test on Windows Vista or later, Macintosh OS X or Linux it must be successfully completed using both a native 32-bit and a native 64-bit data source.

Test Flatbed Scanning

This test sets `CAP_XFERCOUNT` to `0`, `1` and `-1` for a flatbed scanner. It expects an error for the value `0`, and only one image to be transferred per scanning session for the values `1` and `-1`.

1. **Action**: `MSG_RESETALL`

1.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

1.2. **Action**: `MSG_SET CAP_FEEDERENABLED` to `FALSE`

1.3. **Test**: If return is `TWRC_FAILURE / TWCC_BADVALUE`, then scanner does not have a flatbed, proceed to the Test Document Feeder Scanning section

1.4. **Test**: If return is not `TWRC_SUCCESS` and not `TWRC_FAILURE / TWCC_CAPUNSUPPORTED`, end with error

1.5. **Action**: `MSG_SET ICAP_XFERMECH` to `TWSX_NATIVE`

1.5.1. **Test**: If return is not `TWRC_SUCCESS`, end with error

1.6. **Action**: `MSG_SET CAP_XFERCOUNT` to `0`

1.6.1. **Test**: If return code is not `TWRC_FAILURE / TWCC_BADVALUE`, end with an error

1.7. **Action**: `MSG_SET CAP_XFERCOUNT` to `1`

1.7.1. **Test**: If return is not `TWRC_SUCCESS`, end with error

1.8. **Action**: `DG_CONTROL / DAT_USERINTERFACE / MSG_ENABLEDS` with `ShowUI = FALSE` and `ModalUI = FALSE`

1.8.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

1.9. **Action**: Wait for `MSG_XFERREADY`

1.10. **Action**: `DG_IMAGE / DAT_IMAGENATIVEXFER / MSG_GET`

1.10.1. **Test**: If return code is not `TWRC_XFERDONE`, end with an error

1.11. **Action**: `DG_CONTROL / DAT_PENDINGXFERS / MSG_ENDXFER`

1.11.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

1.11.2. **Test**: If `TW_PENDINGXFERS.Count` is not `0`, end with error

1.12. **Action**: `DG_CONTROL / DAT_USERINTERFACE / MSG_DISABLEDS`

1.12.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

1.13. **Action**: `MSG_SET CAP_XFERCOUNT` to `-1`

1.13.1. **Test**: If return is not `TWRC_SUCCESS`, end with error

1.14. **Action**: `DG_CONTROL / DAT_USERINTERFACE / MSG_ENABLEDS` with `ShowUI = FALSE` and `ModalUI = FALSE`

1.14.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

1.15. **Action**: Wait for `MSG_XFERREADY`

1.16. **Action**: `DG_IMAGE / DAT_IMAGENATIVEXFER / MSG_GET`

1.16.1. **Test**: If return code is not `TWRC_XFERDONE`, end with an error

1.17. **Action**: `DG_CONTROL / DAT_PENDINGXFERS / MSG_ENDXFER`

1.17.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

1.17.2. **Test**: If `TW_PENDINGXFERS.Count` is not `0`, end with error

1.18. **Action**: `DG_CONTROL / DAT_USERINTERFACE / MSG_DISABLEDS`

1.18.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

Test Document Feeder Scanning

This test issues multiple image transfer sessions using `DAT_IMAGENATIVEXFER`. It is performed for all available image sources (unspecified, flatbed and/or ADF). Only one image is transferred per session.

2. **Action**: `MSG_RESETALL`

2.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

2.2. **Action**: MSG_SET CAP_FEEDERENABLED to TRUE

2.3. **Test**: If return is TWRC_FAILURE / TWCC_BADVALUE or TWRC_FAILURE / TWCC_CAPUNSUPPORTED, then scanner does not have a Document Feeder, skip the rest of this section

2.4. **Test**: If return is not TWRC_SUCCESS, end with error

2.5. **Action**: MSG_SET ICAP_XFERMECH to TWSX_NATIVE

- 2.5.1. **Test**: If return is not TWRC_SUCCESS, end with error

2.6. **Action**: MSG_SET CAP_XFERCOUNT to 3

- 2.6.1. **Test**: If return is not TWRC_SUCCESS or TWRC_CHECKSTATUS, end with error

2.7. **Action**: MSG_GET CAP_XFERCOUNT

- 2.7.1. **Test**: If return is not TWRC_SUCCESS, end with error
- 2.7.2. **Test**: If value is not equal to 3 do this section
  - 2.7.2.1. **Action**: MSG_SET CAP_XFERCOUNT to 0
    - 2.7.2.1.1. **Test**: If return code is not TWRC_FAILURE / TWCC_BADVALUE, end with an error
  - 2.7.2.2. **Action**: MSG_SET CAP_XFERCOUNT to 1
    - 2.7.2.2.1. **Test**: If return is not TWRC_SUCCESS, end with error
  - 2.7.2.3. **Action**: Ask user to place one sheet of paper in the document feeder
  - 2.7.2.4. DG_CONTROL / DAT_USERINTERFACE / MSG_ENABLEDS with ShowUI = FALSE and ModalUI = FALSE
    - 2.7.2.4.1. **Test**: If return code is not TWRC_SUCCESS, end with an error
  - 2.7.2.5. **Action**: Wait for MSG_XFERREADY
  - 2.7.2.6. **Action**: DG_IMAGE / DAT_IMAGENATIVEXFER / MSG_GET
    - 2.7.2.6.1. **Test**: If return code is not TWRC_XFERDONE, end with an error
  - 2.7.2.7. **Action**: DG_CONTROL / DAT_PENDINGXFERS / MSG_ENDXFER
    - 2.7.2.7.1. **Test**: If return code is not TWRC_SUCCESS, end with an error

2.7.2.7.2. **Test**: If `TW_PENDINGXFERS.Count` is not `0`, end with error

2.7.2.8. **Action**: `DG_CONTROL / DAT_USERINTERFACE / MSG_DISABLEDS`

2.7.2.8.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

2.7.2.9. **Action**: `MSG_SET CAP_XFERCOUNT` to `-1`

2.7.2.9.1. **Test**: If return is not `TWRC_SUCCESS`, end with error

2.7.2.10. **Action**: Ask user to place one sheet of paper in the document feeder

2.7.2.11. **Action**: `DG_CONTROL / DAT_USERINTERFACE / MSG_ENABLEDS` with `ShowUI = FALSE` and `ModalUI = FALSE`

2.7.2.11.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

2.7.2.12. **Action**: Wait for `MSG_XFERREADY`

2.7.2.13. **Action**: `DG_IMAGE / DAT_IMAGENATIVEXFER / MSG_GET`

2.7.2.13.1. **Test**: If return code is not `TWRC_XFERDONE`, end with an error

2.7.2.14. **Action**: `DG_CONTROL / DAT_PENDINGXFERS / MSG_ENDXFER`

2.7.2.14.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

2.7.2.14.2. **Test**: If `TW_PENDINGXFERS.Count` is not `0`, end with error

2.7.2.15. **Action**: `DG_CONTROL / DAT_USERINTERFACE / MSG_DISABLEDS`

2.7.2.15.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

2.7.3. **Test**: If value is equal to `3` do this section

2.7.3.1. **Action**: Ask user to place three sheets of paper in the document feeder

2.7.3.2. **Action**: `MSG_SET CAP_DUPLEXENABLED` to `FALSE`

2.7.3.2.1. **Test**: If return code is not `TWRC_SUCCESS` or `TWRC_FAILURE / TWCC_CAPUNSUPPORTED`, end with error

2.7.3.3. **Action**: `MSG_SET CAP_XFERCOUNT` to `0`

2.7.3.3.1. **Test**: If return code is not `TWRC_FAILURE / TWCC_BADVALUE`, end with an error

2.7.3.4. **Action**: `MSG_SET CAP_XFERCOUNT` to `1`

2.7.3.4.1. **Test**: If return is not `TWRC_SUCCESS`, end with error

2.7.3.5. **Action**: `DG_CONTROL / DAT_USERINTERFACE / MSG_ENABLEDS` with `ShowUI = FALSE` and `ModalUI = FALSE`

2.7.3.5.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

2.7.3.6. **Action**: Wait for `MSG_XFERREADY`

2.7.3.7. **Action**: `DG_IMAGE / DAT_IMAGENATIVEXFER / MSG_GET`

2.7.3.7.1. **Test**: If return code is not `TWRC_XFERDONE`, end with an error

2.7.3.8. **Action**: `DG_CONTROL / DAT_PENDINGXFERS / MSG_ENDXFER`

2.7.3.8.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

2.7.3.8.2. **Test**: If `TW_PENDINGXFERS.Count` is not `0`, end with error

2.7.3.9. **Action**: `DG_CONTROL / DAT_USERINTERFACE / MSG_DISABLEDS`

2.7.3.9.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

2.7.3.10. **Action**: `MSG_SET CAP_XFERCOUNT` to `-1`

2.7.3.10.1. **Test**: If return is not `TWRC_SUCCESS`, end with error

2.7.3.11. **Action**: `DG_CONTROL / DAT_USERINTERFACE / MSG_ENABLEDS` with `ShowUI = FALSE` and `ModalUI = FALSE`

2.7.3.11.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

2.7.3.12. **Action**: Wait for `MSG_XFERREADY`

2.7.3.13. **Action**: `DG_IMAGE / DAT_IMAGENATIVEXFER / MSG_GET`

2.7.3.13.1. **Test**: If return code is not `TWRC_XFERDONE`, end with an error

2.7.3.14. **Action**: `DG_CONTROL / DAT_PENDINGXFERS / MSG_ENDXFER`

2.7.3.14.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

2.7.3.14.2. **Test**: If `TW_PENDINGXFERS.Count` is not `1` or `-1`, end with error

2.7.3.15. **Action**: `DG_IMAGE / DAT_IMAGENATIVEXFER / MSG_GET`

2.7.3.15.1. **Test**: If return code is not `TWRC_XFERDONE`, end with an error

2.7.3.16. **Action**: `DG_CONTROL / DAT_PENDINGXFERS / MSG_ENDXFER`

2.7.3.16.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

2.7.3.16.2. **Test**: If `TW_PENDINGXFERS.Count` is not `0`, end with error

2.7.3.17. **Action**: `DG_CONTROL / DAT_USERINTERFACE / MSG_DISABLEDS`

2.7.3.17.1. **Test**: If return code is not `TWRC_SUCCESS`, end with an error

Post-Test Procedure

When testing is completed, close the data source and the data source manager.

Version Tests

Purpose

Confirm that the data sources responds correctly to different TWAIN versions of data source manager and application.

Pre-Test Procedure

Close the data source manager.

Attempt to scan Multiple Times

Confirm that the data source can respond correctly to different TWAIN version of application and data source manager by attempting to scan using different setups. This tests for hangs and crashes. Use Memory transfer if available. Scan one image in simplex without UI. Testing with old DSM is only for 32-bit data sources only.

1. **Action**: `MSG_OPENDSM` using old DSM as TWAIN version 1.9 application, with `DF_APP2` set,
   - 1.1. **Action**: Attempt to scan
   - 1.2. **Test**: Confirm that the scan succeeds without hanging.
   - 1.3. **Test**: If the application does not receive `MSG_XFERREADY`, then end with error
   - 1.4. **Action**: `MSG_CLOSEDSM`
2. **Action**: `MSG_OPENDSM` using old DSM as TWAIN version 2.x application, with `DF_APP2` not set,
   - 2.1. **Action**: Attempt to scan
   - 2.2. **Test**: Confirm that the scan succeeds without hanging.
   - 2.3. **Test**: If the application does not receive `MSG_XFERREADY`, then end with error
   - 2.4. **Action**: `MSG_CLOSEDSM`
3. **Action**: `MSG_OPENDSM` using old DSM as TWAIN version 2.x application, with `DF_APP2` set,
   - 3.1. **Action**: Attempt to scan
   - 3.2. **Test**: Confirm that the scan succeeds without hanging.
   - 3.3. **Test**: If the application does not receive `MSG_XFERREADY`, then end with error
   - 3.4. **Action**: `MSG_CLOSEDSM`
4. **Action**: `MSG_OPENDSM` using TWAIN 2 DSM as TWAIN version 1.9 application, with `DF_APP2` set,
   - 4.1. **Action**: Attempt to scan
   - 4.2. **Test**: Confirm that the scan succeeds without hanging.
   - 4.3. **Test**: If the application does not receive `MSG_XFERREADY`, then end with error
   - 4.4. **Action**: `MSG_CLOSEDSM`
5. **Action**: `MSG_OPENDSM` using TWAIN 2 DSM as TWAIN version 2.x application, with `DF_APP2` not set,

5.1. **Action**: Attempt to scan

5.2. **Test**: Confirm that the scan succeeds without hanging.

5.3. **Test**: If the application does not receive `MSG_XFERREADY`, then end with error

5.4. **Action**: `MSG_CLOSEDSM`

6. **Action**: `MSG_OPENDSM` using TWAIN 2 DSM as TWAIN version 1.9 application, with `DF_APP2` not set,

6.1. **Action**: Attempt to scan

6.2. **Test**: Confirm that the scan succeeds without hanging.

6.3. **Test**: If the application does not receive `MSG_XFERREADY`, then end with error

6.4. **Action**: `MSG_CLOSEDSM`

Post-Test Procedure

Nothing to do.

Verify Values For MSG_RESETALL and MSG_RESET

Purpose

Confirm that the indicated capabilities have the values required by the Specification after a `DG_CONTROL / DAT_CAPABILITY / MSG_RESETALL` is applied to the entire driver, or a `DG_CONTROL / DAT_CAPABILITY / MSG_RESET` is applied to a single capability.

Pre-Test Procedure

Open the data source manager and the data source that is to be tested.

Test MSG_RESETALL and MSG_RESET

Make sure that `MSG_RESETALL` results in the following values for the indicated capabilities.

1. **Action**: `DG_CONTROL / DAT_CAPABILITY / MSG_RESETALL`

1.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.2. **Action**: `MSG_GETCURRENT ACAP_XFERMECH`

1.2.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.2.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWSX_NATIVE`, then end with error

1.2.3. **Action**: `MSG_RESET ACAP_XFERMECH`

1.2.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.2.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWSX_NATIVE`, then end with error

1.3. **Action**: `MSG_GETCURRENT CAP_AUTHOR`

1.3.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_STRING128`, or the value is not an empty string, then end with error

1.3.3. **Action**: `MSG_RESET CAP_AUTHOR`

1.3.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.3.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_STRING128`, or the value is not an empty string, then end with error

1.4. **Action**: `MSG_GETCURRENT CAP_AUTOFEED`

1.4.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.4.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `TRUE`, then end with error

1.4.3. **Action**: `MSG_RESET CAP_AUTOFEED`

1.4.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.4.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `TRUE`, then end with error

1.5. **Action**: `MSG_GETCURRENT CAP_AUTOMATICCAPTURE`

1.5.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.5.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_INT32`, or the value is not `0`, then end with error

1.5.3. **Action**: `MSG_RESET CAP_AUTOMATICCAPTURE`

1.5.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.5.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_INT32`, or the value is not `0`, then end with error

1.6. **Action**: `MSG_GETCURRENT CAP_CAMERSIDE`

1.6.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.6.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWCS_BOTH`, then end with error

1.6.3. **Action**: `MSG_RESET CAP_CAMERSIDE`

1.6.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.6.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWCS_BOTH`, then end with error

1.7. **Action**: `MSG_GETCURRENT CAP_CAPTION`

1.7.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.7.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_STRING255`, or the value is not an empty string, then end with error

1.7.3. **Action**: `MSG_RESET CAP_CAPTION`

1.7.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.7.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_STRING255`, or the value is not an empty string, then end with error

1.8. **Action**: `MSG_GETCURRENT CAP_CLEARBUFFERS`

1.8.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.8.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWCB_AUTO`, then end with error

1.8.3. **Action**: `MSG_RESET CAP_CLEARBUFFERS`

1.8.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.8.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWCB_AUTO`, then end with error

1.9. **Action**: `MSG_GETCURRENT CAP_CLEARPAGE`

1.9.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.9.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.9.3. **Action**: `MSG_RESET CAP_CLEARPAGE`

1.9.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.9.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.10. **Action**: `MSG_GETCURRENT CAP_DEVICEEVENT`

1.10.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.10.2. **Test**: If the container is not `TW_ARRAY`, or the value is not an empty array, then end with error

1.10.3. **Action**: `MSG_RESET CAP_DEVICEEVENT`

1.10.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.10.3.2. **Test**: If the container is not `TW_ARRAY`, or the value is not an empty array, then end with error

1.11. **Action**: `MSG_GETCURRENT CAP_DOUBLEFEEDDETECTION`

1.11.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.11.2. **Test**: If the container is not `TW_ARRAY`, or the value is not an empty array, then end with error

1.11.3. **Action**: `MSG_RESET CAP_DOUBLEFEEDDETECTION`

1.11.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.11.3.2. **Test**: If the container is not `TW_ARRAY`, or the value is not an empty array, then end with error

1.12. **Action**: `MSG_GETCURRENT CAP_ENDORSER`

1.12.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.12.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT32`, or the value is not `1`, then end with error

1.12.3. **Action**: `MSG_RESET CAP_ENDORSER`

1.12.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.12.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT32`, or the value is not `1`, then end with error

1.13. **Action**: `MSG_GETCURRENT CAP_FEEDERPREP`

1.13.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.13.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.13.3. **Action**: `MSG_RESET CAP_FEEDERPREP`

1.13.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.13.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.14. **Action**: `MSG_GETCURRENT CAP_FEEDPAGE`

1.14.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.14.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.14.3. **Action**: `MSG_RESET CAP_FEEDPAGE`

1.14.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.14.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.15. **Action**: `MSG_GETCURRENT CAP_INDICATORS`

1.15.1. **Test**: If the result is not `TWRC_SUCCESS`, then end with error

1.15.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `TRUE`, then end with error

1.15.3. **Action**: `MSG_RESET CAP_INDICATORS`

1.15.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.15.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `TRUE`, then end with error

1.16. **Action**: `MSG_GETCURRENT CAP_INDICATORS`

1.16.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.16.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `TRUE`, then end with error

1.16.3. **Action**: `MSG_RESET CAP_INDICATORS`

1.16.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.16.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `TRUE`, then end with error

1.17. **Action**: `MSG_GETCURRENT CAP_JOBCONTROL`

1.17.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.17.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWJC_NONE`, then end with error

1.17.3. **Action**: `MSG_RESET CAP_JOBCONTROL`

1.17.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.17.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWJC_NONE`, then end with error

1.18. **Action**: `MSG_GETCURRENT CAP_MICRENABLED`

1.18.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.18.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.18.3. **Action**: `MSG_RESET CAP_MICRENABLED`

1.18.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.18.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.19. **Action**: `MSG_GETCURRENT CAP_PAPERHANDLING`

1.19.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.19.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWPH_NORMAL`, then end with error

1.19.3. **Action**: `MSG_RESET CAP_PAPERHANDLING`

1.19.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.19.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWPH_NORMAL`, then end with error

1.20. **Action**: `MSG_GETCURRENT CAP_PRINTERENABLED`

1.20.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.20.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.20.3. **Action**: `MSG_RESET CAP_PRINTERENABLED`

1.20.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.20.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.21. **Action**: `MSG_GETCURRENT CAP_PRINTERINDEX`

1.21.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.21.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT32`, or the value is not `1`, then end with error

1.21.3. **Action**: `MSG_RESET CAP_PRINTERINDEX`

1.21.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.21.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT32`, or the value is not `1`, then end with error

1.22. **Action**: `MSG_GETCURRENT CAP_REACQUIREALLOWED`

1.22.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.22.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.22.3. **Action**: `MSG_RESET CAP_REACQUIREALLOWED`

1.22.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.22.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.23. **Action**: `MSG_GETCURRENT CAP_SEGMENTED`

1.23.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.23.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWSG_NONE`, then end with error

1.23.3. **Action**: `MSG_RESET CAP_SEGMENTED`

1.23.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.23.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWSG_NONE`, then end with error

1.24. **Action**: `MSG_GETCURRENT CAP_TIMEBEFOREFIRSTCAPTURE`

1.24.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.24.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_INT32`, or the value is not `0`, then end with error

1.24.3. **Action**: `MSG_RESET CAP_TIMEBEFOREFIRSTCAPTURE`

1.24.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.24.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_INT32`, or the value is not `0`, then end with error

1.25. **Action**: `MSG_GETCURRENT CAP_TIMEBETWEENCAPTURES`

1.25.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.25.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_INT32`, or the value is not `0`, then end with error

1.25.3. **Action**: `MSG_RESET CAP_TIMEBETWEENCAPTURES`

1.25.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.25.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_INT32`, or the value is not `0`, then end with error

1.26. **Action**: `MSG_GETCURRENT CAP_THUMBNAILSENABLED`

1.26.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.26.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.26.3. **Action**: `MSG_RESET CAP_THUMBNAILSENABLED`

1.26.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.26.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.27. **Action**: `MSG_GETCURRENT CAP_XFERCOUNT`

1.27.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.27.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_INT16`, or the value is not `-1`, then end with error

1.27.3. **Action**: `MSG_RESET CAP_XFERCOUNT`

1.27.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.27.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_INT16`, or the value is not `-1`, then end with error

1.28. **Action**: `MSG_GETCURRENT ICAP_AUTOBRIGHT`

1.28.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.28.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.28.3. **Action**: `MSG_RESET ICAP_AUTOBRIGHT`

1.28.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.28.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.29. **Action**: `MSG_GETCURRENT ICAP_AUTODISCARDBLANKPAGES`

1.29.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.29.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWBP_DISABLED`, then end with error

1.29.3. **Action**: `MSG_RESET ICAP_AUTODISCARDBLANKPAGES`

1.29.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.29.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWBP_DISABLED`, then end with error

1.30. **Action**: `MSG_GETCURRENT ICAP_AUTOMATICCOLORENABLED`

1.30.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

  - 1.30.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error
  - 1.30.3. **Action**: `MSG_RESET ICAP_AUTOMATICCOLORENABLED`
    - 1.30.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error
    - 1.30.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

- 1.31. **Action**: `MSG_GETCURRENT ICAP_AUTOMATICCOLORNONCOLORPIXELTYPE`
  - 1.31.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability
  - 1.31.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWPT_BW`, then end with error
  - 1.31.3. **Action**: `MSG_RESET ICAP_AUTOMATICCOLORNONCOLORPIXELTYPE`
    - 1.31.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error
    - 1.31.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWPT_BW`, then end with error

- 1.32. **Action**: `MSG_GETCURRENT ICAP_AUTOMATICROTATE`
  - 1.32.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability
  - 1.32.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error
  - 1.32.3. **Action**: `MSG_RESET ICAP_AUTOMATICROTATE`
    - 1.32.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error
    - 1.32.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

- 1.33. **Action**: `MSG_GETCURRENT ICAP_AUTOSIZE`
  - 1.33.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability
  - 1.33.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWAS_NONE`, then end with error
  - 1.33.3. **Action**: `MSG_RESET ICAP_AUTOSIZE`
    - 1.33.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error
    - 1.33.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWAS_NONE`, then end with error

- 1.34. **Action**: `MSG_GETCURRENT ICAP_BARCODEDETECTIONENABLED`
  - 1.34.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.34.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.34.3. **Action**: `MSG_RESET ICAP_BARCODEDETECTIONENABLED`

1.34.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.34.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.35. 1.35 **Action**: `MSG_GETCURRENT ICAP_BITORDER`

1.35.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.35.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWBO_MSBFIRST`, then end with error

1.35.3. **Action**: `MSG_RESET ICAP_BITORDER`

1.35.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.35.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWBO_MSBFIRST`, then end with error

1.36. **Action**: `MSG_GETCURRENT ICAP_BITORDERCODES`

1.36.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.36.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWBO_LSBFIRST`, then end with error

1.36.3. **Action**: `MSG_RESET ICAP_BITORDERCODES`

1.36.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.36.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWBO_LSBFIRST`, then end with error

1.37. **Action**: `MSG_GETCURRENT ICAP_BRIGHTNESS`

1.37.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.37.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `0`, then end with error

1.37.3. **Action**: `MSG_RESET ICAP_BRIGHTNESS`

1.37.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.37.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `0`, then end with error

1.38. **Action**: `MSG_GETCURRENT ICAP_CCITTKFACTOR`

1.38.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.38.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `4`, then end with error

1.38.3. **Action**: `MSG_RESET ICAP_CCITTKFACTOR`

1.38.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.38.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `4`, then end with error

1.39. **Action**: `MSG_GETCURRENT ICAP_COLORMANAGEMENTENABLED`

1.39.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.39.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `TRUE`, then end with error

1.39.3. **Action**: `MSG_RESET ICAP_COLORMANAGEMENTENABLED`

1.39.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.39.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `TRUE`, then end with error

1.40. **Action**: `MSG_GETCURRENT ICAP_COMPRESSION`

1.40.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.40.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWCP_COMPRESSION`, then end with error

1.40.3. **Action**: `MSG_RESET ICAP_COMPRESSION`

1.40.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.40.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWCP_COMPRESSION`, then end with error

1.41. **Action**: `MSG_GETCURRENT ICAP_CONTRAST`

1.41.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.41.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `0`, then end with error

1.41.3. **Action**: `MSG_RESET ICAP_CONTRAST`

1.41.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.41.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `0`, then end with error

1.42. **Action**: `MSG_GETCURRENT ICAP_EXTIMAGEINFO`

1.42.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.42.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `TRUE`, then end with error

1.42.3. **Action**: `MSG_RESET ICAP_EXTIMAGEINFO`

1.42.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.42.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `TRUE`, then end with error

1.43. **Action**: `MSG_GETCURRENT ICAP_FILTER`

1.43.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.43.2. **Test**: If the container is not `TW_ARRAY`, or the value is not an empty array, then end with error

1.43.3. **Action**: `MSG_RESET ICAP_FILTER`

1.43.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.43.3.2. **Test**: If the container is not `TW_ARRAY`, or the value is not an empty array, then end with error

1.44. **Action**: `MSG_GETCURRENT ICAP_FLIPROTATION`

1.44.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.44.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWFR_BOOK`, then end with error

1.44.3. **Action**: `MSG_RESET ICAP_FLIPROTATION`

1.44.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.44.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWFR_BOOK`, then end with error

1.45. **Action**: `MSG_GETCURRENT ICAP_GAMMA`

1.45.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.45.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `2.2`, then end with error

1.45.3. **Action**: `MSG_RESET ICAP_GAMMA`

1.45.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.45.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `2.2`, then end with error

1.46. **Action**: `MSG_GETCURRENT ICAP_HIGHLIGHT`

1.46.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.46.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `255`, then end with error

1.46.3. **Action**: `MSG_REEST ICAP_HIGHLIGHT`

1.46.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.46.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `255`, then end with error

1.47. **Action**: `MSG_GETCURRENT ICAP_IMAGEMERGE`

1.47.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.47.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWIM_NONE`, then end with error

1.47.3. **Action**: `MSG_RESET ICAP_IMAGEMERGE`

1.47.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.47.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWIM_NONE`, then end with error

1.48. **Action**: `MSG_GETCURRENT ICAP_IMAGEMERGEHEIGHTTHRESHOLD`

1.48.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.48.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `0`, then end with error

1.48.3. **Action**: `MSG_GETCURRENT ICAP_IMAGEMERGEHEIGHTTHRESHOLD`

1.48.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.48.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `0`, then end with error

1.49. **Action**: `MSG_GETCURRENT ICAP_MIRROR`

1.49.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.49.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWMR_NONE`, then end with error

1.49.3. **Action**: `MSG_RESET ICAP_MIRROR`

1.49.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.49.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWMR_NONE`, then end with error

1.50. **Action**: `MSG_GETCURRENT ICAP_ORIENTATION`

1.50.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.50.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWOR_PORTRAIT`, then end with error

1.50.3. **Action**: `MSG_RESET ICAP_ORIENTATION`

1.50.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.50.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWOR_PORTRAIT`, then end with error

1.51. **Action**: `MSG_GETCURRENT ICAP_OVERSCAN`

1.51.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.51.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWOV_NONE`, then end with error

1.51.3. **Action**: `MSG_RESET ICAP_OVERSCAN`

1.51.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.51.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWOV_NONE`, then end with error

1.52. **Action**: `MSG_GETCURRENT ICAP_PATCHCODEDETECTIONENABLED`

1.52.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.52.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.52.3. **Action**: `MSG_RESET ICAP_PATCHCODEDETECTIONENABLED`

1.52.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.52.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.53. **Action**: `MSG_GETCURRENT ICAP_PIXELFLAVOR`

1.53.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.53.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWPF_CHOCOLATE`, then end with error

1.53.3. **Action**: `MSG_RESET ICAP_PIXELFLAVOR`

1.53.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.53.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWPF_CHOCOLATE`, then end with error

1.54. **Action**: `MSG_GETCURRENT ICAP_PIXELFLAVORCODES`

1.54.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.54.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWPF_CHOCOLATE`, then end with error

1.54.3. **Action**: `MSG_RESET ICAP_PIXELFLAVORCODES`

1.54.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.54.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWPF_CHOCOLATE`, then end with error

1.55. **Action**: `MSG_GETCURRENT ICAP_ROTATION`

1.55.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.55.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `0`, then end with error

1.55.3. **Action**: `MSG_RESET ICAP_ROTATION`

1.55.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.55.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `0`, then end with error

1.56. **Action**: `MSG_GETCURRENT ICAP_SHADOW`

1.56.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.56.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `0`, then end with error

1.56.3. **Action**: `MSG_RESET ICAP_SHADOW`

1.56.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.56.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `0`, then end with error

1.57. **Action**: `MSG_GETCURRENT ICAP_THRESHOLD`

1.57.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.57.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `128`, then end with error

1.57.3. **Action**: `MSG_RESET ICAP_THRESHOLD`

1.57.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.57.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `128`, then end with error

1.58. **Action**: `MSG_GETCURRENT ICAP_TILES`

1.58.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.58.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.58.3. **Action**: `MSG_RESET ICAP_TILES`

1.58.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.58.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.59. **Action**: `MSG_GETCURRENT ICAP_TIMEFILL`

1.59.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.59.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `1`, then end with error

1.59.3. **Action**: `MSG_RESET ICAP_TIMEFILL`

1.59.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.59.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `1`, then end with error

1.60. **Action**: `MSG_GETCURRENT ICAP_UNDEFINEDIMAGESIZE`

1.60.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.60.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.60.3. **Action**: `MSG_RESET ICAP_UNDEFINEDIMAGESIZE`

1.60.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.60.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_BOOL`, or the value is not `FALSE`, then end with error

1.61. **Action**: `MSG_GETCURRENT ICAP_UNITS`

1.61.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.61.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWUN_INCHES`, then end with error

1.61.3. **Action**: `MSG_RESET ICAP_UNITS`

1.61.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.61.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `TWUN_INCHES`, then end with error

1.62. **Action**: `MSG_GETCURRENT ICAP_XFERMECH`

1.62.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.62.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `-1`, then end with error

1.62.3. **Action**: `MSG_RESET ICAP_XFERMECH`

1.62.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.62.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_UINT16`, or the value is not `-1`, then end with error

1.63. **Action**: `MSG_GETCURRENT ICAP_XSCALING`

1.63.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.63.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `1`, then end with error

1.63.3. **Action**: `MSG_RESET ICAP_XSCALING`

1.63.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.63.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `1`, then end with error

1.64. **Action**: `MSG_GETCURRENT ICAP_YSCALING`

1.64.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.64.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `1`, then end with error

1.64.3. **Action**: `MSG_RESET ICAP_YSCALING`

1.64.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.64.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_FIX32`, or the value is not `1`, then end with error

1.65. **Action**: `MSG_GETCURRENT ICAP_ZOOMFACTOR`

1.65.1. **Test**: If result is not `TWRC_SUCCESS`, then skip down to the next capability

1.65.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_INT16`, or the value is not `0`, then end with error

1.65.3. **Action**: `MSG_RESET ICAP_ZOOMFACTOR`

1.65.3.1. **Test**: If result is not `TWRC_SUCCESS`, then end with error

1.65.3.2. **Test**: If the container is not `TW_ONEVALUE`, or the data type is not `TWTY_INT16`, or the value is not `0`, then end with error