TWAIN Errata

For Version 2.4

June 9th, 2017

Purpose

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

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

History

Date	Comment
July 12th, 2016	Typos

Notes

Notes
All page numbers refer to the latest "TWAIN 2.3 Spec.pdf" document.
The [Technical Notes] section is where comments are made to clarify the decisions made by the Technical Committee, these comments are not added to the Specification.
The [Update Instructions] sections guide the Tech Writer in what changes are to be made to the Specification.
All titles are tagged in the following way: [discuss] – item needs further discussion before review [review] – item is under review before being ready or more discussion [ready] – item has been reviewed and is ready to go into the Specification [done] – item has been correctly added to the Specification [removed] – item has been removed (it will not go into any Specification) [defer] – item has been deferred to a future version of the Specification

Contents

Typos [REVIEW]

[Update Instructions]
Page 9-8 (PDF page 406) change " TWP= CH_PATCH6" to “TWPCH_PATCH6"

Allowed Values: `TWPCH_PATCH1, TWPCH_PATCH2, TWPCH_PATCH3,`

`TWPCH_PATCH4, TWP= CH_PATCH6, TWPCH_PATCHT`

Numeric types [REVIEW]

[Update Instructions]
Page 8-6 (PDF page 300) change the following section:

Numeric types

```
typedef char TW_INT8, FAR *pTW_INT8;
typedef short TW_INT16, FAR *pTW_INT16;
#if defined(__APPLE__) /* cf: Mac version of TWAIN.h */
    typedef int TW_INT32, FAR *pTW_INT32;
#else
    typedef int TW_INT32, FAR *pTW_INT32;
#endif
typedef unsigned char TW_UINT8, FAR *pTW_UINT8;
typedef unsigned short TW_UINT16, FAR *pTW_UINT16;
#if defined(__APPLE__) /* cf: Mac version of TWAIN.h */
    typedef unsigned int TW_UINT32, FAR *pTW_UINT32;
#else
    typedef unsigned long TW_UINT32, FAR *pTW_UINT32;
#endif
typedef unsigned short TW_BOOL, FAR *pTW_BOOL;
```

To be as follows:

Numeric types

```
// See specification: Change to TW_INT32 and TW_UINT32 on 64-bit Linux
typedef char TW_INT8, FAR *pTW_INT8;
typedef short TW_INT16, FAR *pTW_INT16;
#if defined(__APPLE__) /* cf: Mac version of TWAIN.h */
    typedef int TW_INT32, FAR *pTW_INT32;
#elif TW_LEGACY_INTLONG
    typedef long TW_INT32, FAR *pTW_INT32;
#else
    typedef int TW_INT32, FAR *pTW_INT32;
#endif
typedef unsigned char TW_UINT8, FAR *pTW_UINT8;
typedef unsigned short TW_UINT16, FAR *pTW_UINT16;
#if defined(__APPLE__) /* cf: Mac version of TWAIN.h */
    typedef unsigned int TW_UINT32, FAR *pTW_UINT32;
#elif TW_LEGACY_INTLONG
    typedef unsigned long TW_UINT32, FAR *pTW_UINT32;
#else
    typedef unsigned int TW_UINT32, FAR *pTW_UINT32;
#endif
typedef unsigned short TW_BOOL, FAR *pTW_BOOL;
```

Change to TW_INT32 and TW_UINT32 on 64-bit Linux [REVIEW]

[Update Instructions]
Page 12-14 (PDF page 654) add the following after the section on "File Transfer"

Change to TW_INT32 and TW_UINT32 on 64-bit Linux

The TWAIN Specification defines TW_INT32 and TW_UINT32 as "long" for Windows and Linux. This is not a problem on Windows, because it follows LLP64, which defines "long" as a 32-bit integer, equivalent to an "int"

Linux, however, conforms to LP64, which defines "long" as a 64-bit integer.

This means that on 64-bit Linux systems TW_INT32 and TW_UINT32 are 64-bit integers. This isn't a problem in terms of TWAIN communication, but it's a serious problem when these data types are used in other operations.

Therefore, beginning with the release of version 2.4 of the Data Source Manager, applications, the data source manager, and TWAIN drivers, must all use the version of twain.h that specifies that TW_INT32 is an "int" and TW_UINT32 is an "unsigned int".

The TWAIN Working Group feels that this move is necessary. It also feels that this move is possible because adoption of TWAIN on 64-bit Linux systems is still in its early stages.

There is no intention to try to support systems that mix old drivers with new drivers. This is because older applications and data source managers have no way of knowing about this change or dealing with it, and because the change in the size of structures such as TW_IDENTITY make it likely that programs will crash.