

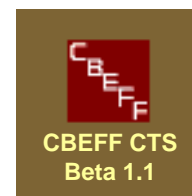
**NIST/ITL Conformance Test Suite for Patron Format A
Data Structures Specified in ANSI INCITS 398-2008,
Common Biometric Exchange Formats Framework
(CBEFF)**

Beta Implementation V1.1

User Guide
August 2008

Fernando Podio, NIST/ITL, CTS Development Project Manager
Yooyoung Lee, Department of Computer Engineering,
University of Chung-Ang, Korea (NIST Guest Researcher)
Mark Jerde, ID Technology Partners (NIST Contractor)
Dylan Yaga, NIST STEP Student

**National Institute of Standards and Technology (NIST)
Information Technology Laboratory (ITL)
Computer Security Division (CSD)**



DISCLAIMER FOR

NIST/ITL CONFORMANCE TEST SUITE (CTS) FOR PATRON FORMAT A DATA STRUCTURES SPECIFIED IN ANSI INCITS 398-2008, COMMON BIOMETRIC EXCHANGE FORMATS FRAMEWORK (CBEFF)

BETA IMPLEMENTATION V1.1

August 21, 2008

The software was developed by the National Institute of Standards and Technology (NIST), an agency of the Federal Government. Pursuant to Title 15 United States Code Section 105, works of NIST are not subject to copyright protection in the United States and are considered to be in the public domain. Thus, the software may be freely reproduced and used. Please explicitly acknowledge the National Institute of Standards and Technology as the source of the software.

This software is released by NIST as a service and is expressly provided "AS IS." NIST MAKES NO WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT AND DATA ACCURACY. NIST DOES NOT REPRESENT OR WARRANT THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT ANY DEFECTS WILL BE CORRECTED.

NIST does not warrant or make any representations regarding the use of the software or the results thereof, including but not limited to the correctness, accuracy, reliability or usefulness of the software. By using this software or by incorporating this software into another product, you agree to hold harmless the United States Government for any and all damages or liabilities that arise out of such use.

Certain trade names and company products are mentioned in the text or identified. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the products are necessarily the best available for the purpose. With the exception of material marked as copyrighted, information presented in this document is considered public information and may be distributed or copied. Use of appropriate byline/photo/image credits is requested..

This work was funded by NIST/ITL CSD and the NIST/ITL Identity Management Systems Program. It was funded in part by the National Institute of Justice.

Contents

1. Introduction	4
1.1. Concepts on Conformance Testing	4
2. Requirements	5
2.1 Configuration	5
2.2 CBEFF CTS Beta 1.1 installation	5
2.2.1 Location of the Sample Data	6
2.2.2 Location of the Log and Report Files	6
2.3 CBEFF_CTS Removal	6
2.4 Starting the CBEFF CTS	6
3. CTS Operating Instructions	6
3.1 CBEFF CTS Welcome Window	7
3.1.1 Title Bar	7
3.1.2 Menu Bar	7
3.1.3 Log File Prefix and Log Filepath Text Boxes	8
3.1.4 TestCase/Manifest/BinaryFile Editor	8
3.1.5 Test Binary Files	8
3.1.6 Check Test Cases	8
3.2 Test Case/Manifest/Binary File Editor Window	9
3.2.1 Title Bar	9
3.2.2 Menu Bar	9
3.2.3 Icon Bar	13
3.2.4 Name (Manifest/Test Case) and Description Panel	14
3.2.5 Expected Result Panel	14
3.2.6 Fields and Binary Data Panel	16
3.2.7 Test Panel	17
3.2.8 Binary Data Panel	18
3.2.9 Developing Manifests, Test Cases and Binary Files	18
3.3 Test Binary Files Window	23
3.3.1 Title Bar	24
3.3.2 Menu Bar	24
3.3.3 Icon bar	25
3.3.4 Manifest Open panel	26
3.3.5 Testing Binary Files	26
3.3.6 Test Status Panel	27
3.4 Check Test Cases Window	28
3.4.1 Title Bar	28
3.4.2 Menu Bar	28
3.4.3 Icon Bar	29
3.4.4 Test Case Panel	29
3.4.5 Test Status Panel	30
3.4.6 Test Case Analysis	30
Annex A. List of Error Messages	31
Annex B. Test Log Displaying “All Errors” Information	35
Annex C. Test Report Displaying “All Errors” Information	36
Annex D. IBIA Identifiers	37

NIST/ITL Conformance Test Suite for Patron Format A (PFA) Data Structures Specified in ANSI INCITS 398-2008, Common Biometric Exchange Formats Framework (CBEFF)

**Beta Implementation V1.1
August 21, 2008**

Users Guide

1. Introduction

This document contains computer requirements, installation instructions and CBEFF Patron Format A Conformance Test Suite (CTS) operation instructions. Two companion documents are available: “Quick Start Guide” and “Overview”.

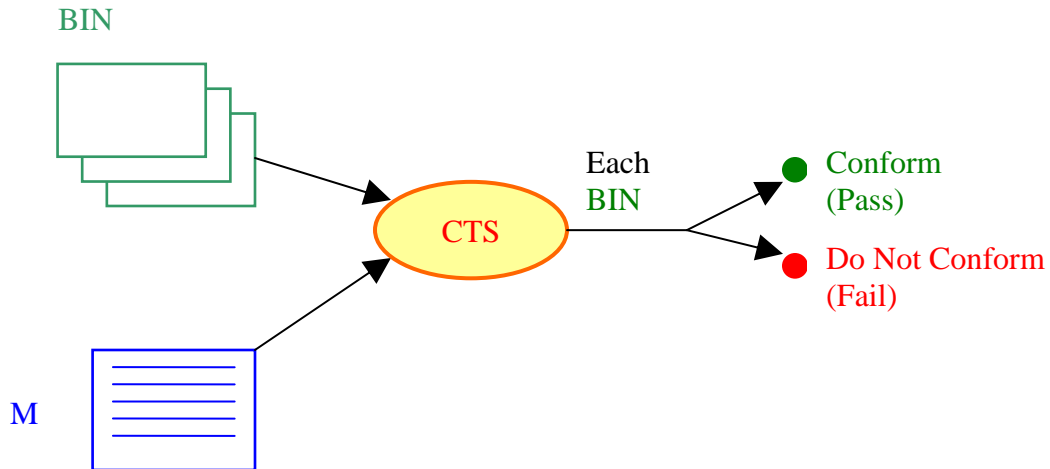
The objective of the “Quick Start Guide” is to get the operator familiarized with key CTS features in a very short time. The “Overview” document provides a brief background on NIST/ITL/CSD support for the development of biometric standards and related technology development efforts. It discusses the need for conformance testing and conformance testing methodology standards development in support of biometric interoperability and data interchange and describes the CBEFF PFA CTS architecture and its functionality.

1.1. Concepts on Conformance Testing

Standards-based conformance testing tools (called Conformance Test Suites in this document) help both developers and users by validating conformance claims, leading to greatly increased levels of confidence in products.

The CTS for Patron Format A data structures specified in ANSI INCITS 398-2008 was developed to determine whether one or more binary file implementations of Biometric Information Records (BIRs) based on this Patron Format conform or not to the standard.

As shown in the diagram below, these binary files are tested against a “Manifest”. A Manifest is a file that defines the format that the binary files must conform to. See the “Overview” document for more information on the required Manifest characteristics.



The CTS uses Test Cases to ensure it operates correctly. A Test Case is a file that contains a Manifest, Binary Data, and the Expected CTS Test Result Information. The Expected Result can be either “Pass” or “Fail for a Specific Reason”. Test Cases are primarily used by the CTS developers. See the “Overview” document for more information on Test Cases. Section 3.2.9.3 of this document discusses how to test and develop Test Cases.

2. Requirements

2.1 Configuration

The CTS has been tested with these operating systems:

- Microsoft Windows® XP Professional Edition
- Microsoft Windows® Vista Ultimate

Recommended minimum display resolution: 1280 X1024 pixels

2.2 CBEFF CTS Beta 1.1 installation

Administrative rights may be required.

- 1) Download [Setup CBEFF CTS Beta 1.1.exe](#) from the CBEFF Patron Format A CTS website:
- 2) Double-click [Setup CBEFF CTS Beta 1.1.exe](#)
- 3) Follow the installation program instructions.

2.2.1 Location of the Sample Data

The samples files are installed where all computer users can access them.

In Microsoft Windows XP Professional the default location is

C:\Documents and Settings\All Users\Shared Documents\CBEFF CTS Beta 1.1

In Microsoft Windows Vista Ultimate the default location is

C:\Users\Public\Documents\CBEFF CTS Beta 1.1

2.2.2 Location of the Log and Report Files

By default Log and Report Files are saved to the folder

[My Documents]\CBEFF CTS Beta 1.1\Logs

The user can change the folder on the CTS Welcome Window.

2.3 CBEFF_CTS Removal

Administrative rights may be required.

To remove the CBEFF_CTS use Add and Remove Programs in the Control Panel.

2.4 Starting the CBEFF CTS

Select the CBEFF CTS program from the Windows Start menu. The default location is Start | All Programs (or Programs) | CBEFF CTS Beta 1.1 | CBEFF CTS Beta 1.1.

3. CTS Operating Instructions

This Section provides detailed descriptions of all CTS functions, windows, window elements, and main dialogs.

3.1 CBEFF CTS Welcome Window



3.1.1 Title Bar

The title bar displays the CTS name and version number.

3.1.2 Menu Bar

Includes the File and Help menus.

File Menu

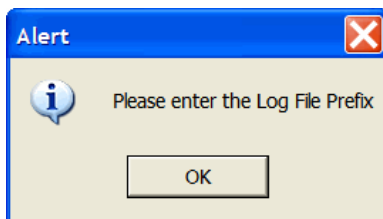
Select Exit to close the CBEFF_CTS.

Help menu

Displays a Disclaimer and the Contributors' list (Disclaimer option) and the CTS version (About option)

3.1.3 Log File Prefix and Log Filepath Text Boxes

Enter at least one character (maximum three) in the Log File Prefix text box and click the Apply button before selecting any testing/editing option. This prefix will be the first part of the Test Logs and Test Reports file names. (See section 3.2.2 - Result menu.) This prefix can be changed at any time. The last entry will be used in the file names. This message is displayed if the prefix field is empty when one of the main window testing/editing options is selected.



To change the folder in which the Logs and Reports are saved either enter the folder's name in the Log File path text box or click the Browse ("...") button, then click the Apply button.

The three main CTS windows accessed from the Welcome window are:

3.1.4 TestCase/Manifest/BinaryFile Editor

Select this button (**#1**) in the Welcome window to generate a Manifest, Test Case or Binary file. As stated above, the Manifest is a file that defines the format to which binary files must conform.

This window can also display, edit, test or save files (see Section 3.2).

3.1.5 Test Binary Files

Select this button (**#2**) to perform conformance testing for Binary files against a Manifest (see Section 3.3). This Manifest could be the sample provided with the CTS, developed by a third party or previously developed by the operator using the CTS or other means.

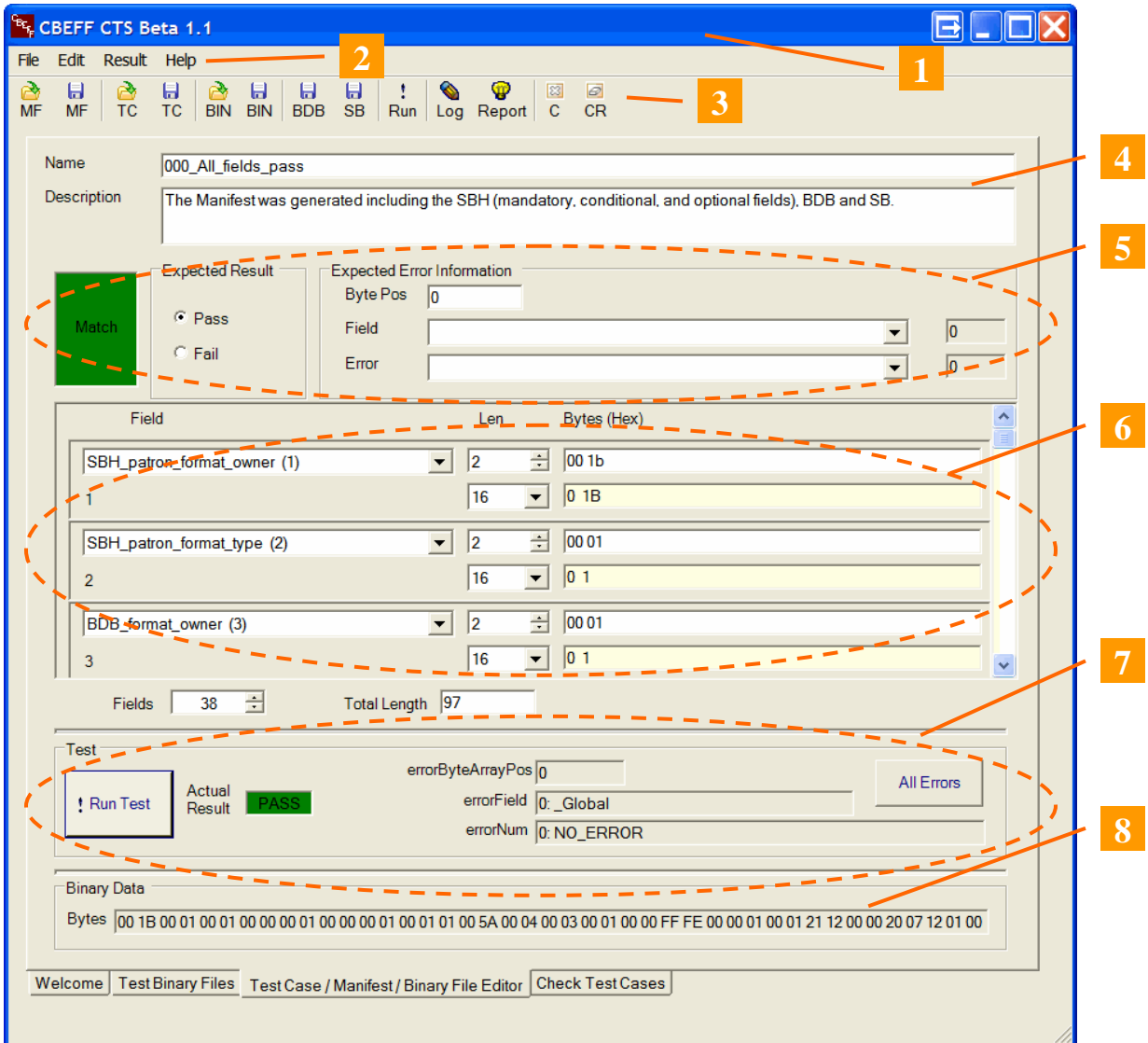
3.1.6 Check Test Cases

Select this button (**#3**) to test one or more Test Cases (see Section 3.4).

These windows can also be accessed by clicking the tabs (#4) at the bottom of the windows.

3.2 Test Case/Manifest/Binary File Editor Window

Manifests, Test Cases and Binary Files are developed using this window.



3.2.1 Title Bar

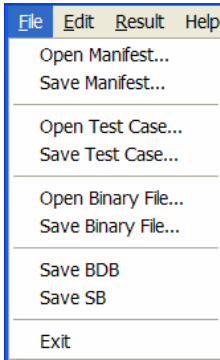
The Title bar (#1) displays the CBEFF CTS name and version number.

3.2.2 Menu Bar

The Menu bar (#2) includes the following options: File, Edit, Result and Help.

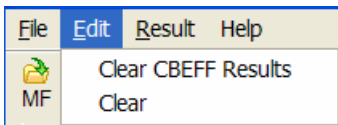
File Menu

The file menu allows the operator to Open/Save a Manifest or Test Case (in XML format), Open/Save a Binary File and Save a Biometric Data Block or a Security Block (as binary data).



Exit closes the CTS

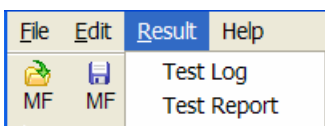
Edit menu



Select Clear CBEFF Results to clear the test results.

Select Clear to erase all window content.

Result menu



Select Test Log to display the Test Log

Select Test Report to display the Test Report.

If a test has not been run a warning message will be shown: "A test must be run first before selecting Test Log/Test Report".

The output of a test is saved as both a Test Log and a Test Report. The Test Log (XML) file is provided for future automated data processing such as importing the results in a data base. The HTML Test Report provides the same information as the Test Log, formatted in human readable form.

The Test Logs and Test Reports are saved into the default or user specified folder. They have the following file name format:

“Log File Prefix_398_2008_PFA_YYYYMMDD_hhmmss_RandomNumber”

- Log File Prefix: The one-to-three character field entered in the Prefix window of the Welcome page.
- 398_2008_PFA: Abbreviation of ANSI INCITS 398-2008 Patron Format A.
- YYYYMMDD: Date as year, month and day.
- hhmmss: Time as hour, minute and second.
- Random Number: A three-digit random value

For example, the Test Log file name could be:

“YYL_398_2008_PFA_2008-04-05_154434-718.xml”

The Test Report file name could be:

“YYL_398_2008_PFA_2008-04-05_154612-656.html”.

▪ **Test Log option**

Select this option to view the Test Log. The Test Log includes the elements shown in this table. A screenshot of this Test Log is shown below.

File Type	Manifest, Test Case or Binary (M/TC/B) file
Patron Format	In this CTS Beta version: ANSI INCITS 398-2008 – Patron Format A
Date/Time	Run time (UTC) of the test run.
Name	Name of the M/TC opened in the window
Description	Description of the M/TC opened in the window to test a M/TC or a Binary file
File Name	Name of the file that was tested (M/TC/B)
Expected Result	Can be specified by the user in the Editor window before running the test.
Number of Fields	Number of fields tested in the record.
Field Sequence	Field order determined from the file.
Field Name	Name of the field

Defined Length	Length specified in the Manifest
Actual Length	Length of the Binary data which was read as the Actual length
BytesBase64	MIME base 64 formatted binary data.
BytesHex	Hexadecimal representation of the binary data
Test Result verdict	Statement on whether the actual test result is the same as the Expected Result

```

<?xml version="1.0" encoding="UTF-8" ?>
- <TestSuite Type="Manifest/TestCases" DLL="INCITS 398-2008 - Patron Format A"
  DateTime="5/22/2008 3:03:20 PM UTC" NumTestCases="1" NumPass="1" NumFail="0">
- <TestCase xsi:noNamespaceSchemaLocation="Schema_03.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Name>000_All_fields_pass</Name>
  <Description>The Manifest was generated including the SBH (mandatory, conditional, and
  optional fields), BDB and SB.</Description>
  <FileName>000_All_fields_pass.xml</FileName>
  <ExpectedResult>Pass</ExpectedResult>
  <NumberOfFields>38</NumberOfFields>
  <Field Sequence="1" Name="SBH_patron_format_owner" DefinedLength="2"
  ActualLength="2" BytesBase64="ABs=" Hex="001B" />
  <Field Sequence="2" Name="SBH_patron_format_type" DefinedLength="2" ActualLength="2"
  BytesBase64="AAE=" Hex="0001" />
  <Field Sequence="3" Name="BDB_format_owner" DefinedLength="2" ActualLength="2"
  BytesBase64="AAE=" Hex="0001" />
  <Field Sequence="4" Name="BDB_format_type" DefinedLength="2" ActualLength="2"
  BytesBase64="AAA=" Hex="0000" />
  <Field Sequence="5" Name="Product_format_owner" DefinedLength="2" ActualLength="2"
  BytesBase64="AAE=" Hex="0001" />
  <Field Sequence="6" Name="Product_format_type" DefinedLength="2" ActualLength="2"
  BytesBase64="AAA=" Hex="0000" />
  <Field Sequence="7" Name="Device_format_owner" DefinedLength="2" ActualLength="2"
  BytesBase64="AAE=" Hex="0001" />
  <Field Sequence="8" Name="Device_format_type" DefinedLength="2" ActualLength="2"
  BytesBase64="AAE=" Hex="0001" />
  <Field Sequence="9" Name="Record_attributes" DefinedLength="1" ActualLength="1"
  BytesBase64="AQ==" Hex="01" />
  <Field Sequence="10" Name="SBH_length" DefinedLength="2" ActualLength="2"
  BytesBase64="AFo=" Hex="005A" />
  <Field Sequence="11" Name="BDB_length" DefinedLength="2" ActualLength="2"
  BytesBase64="AAQ=" Hex="0004" />

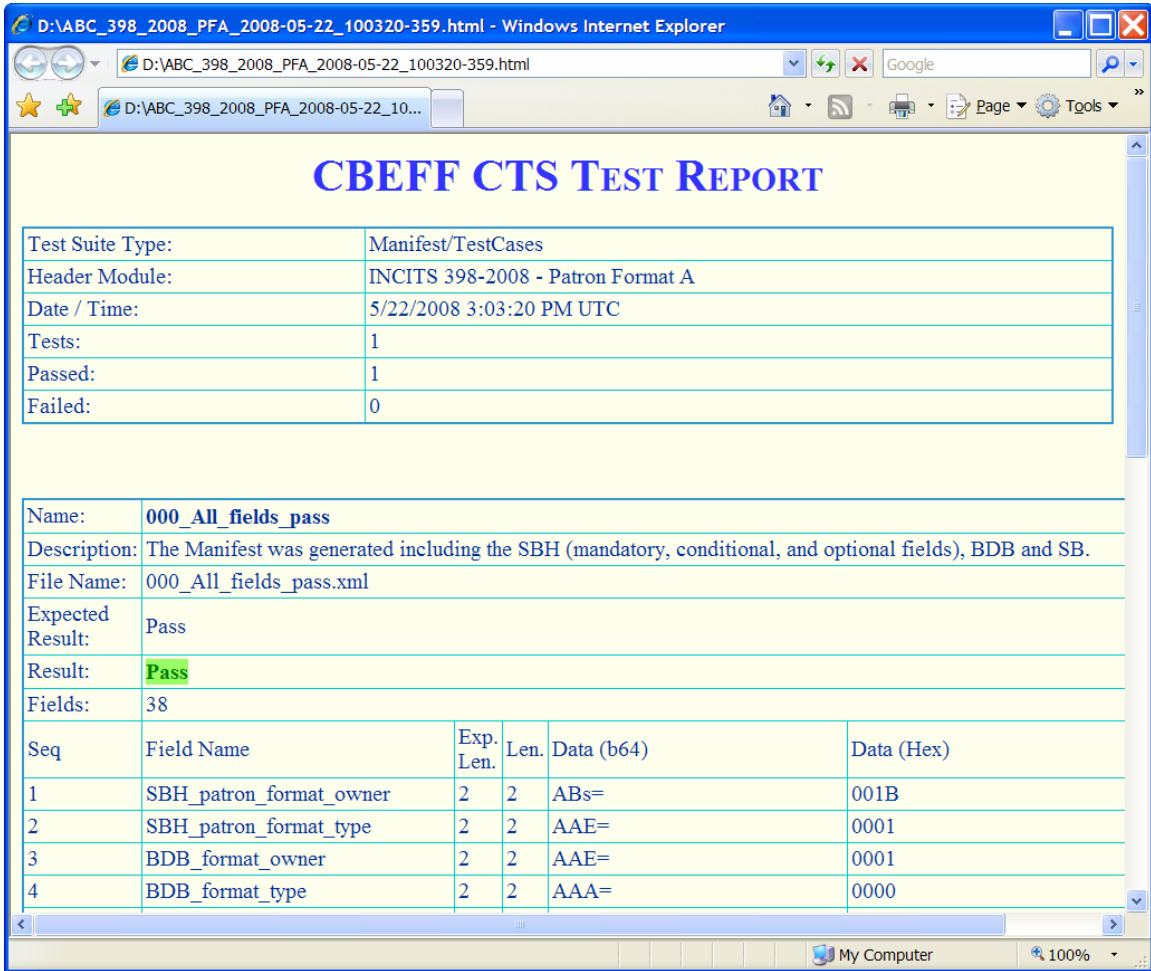
```

▪ **Test Report Option**

Select this option to view the Test Report. A screenshot of this Test Report is shown below.

Most of the fields in the format are self-explanatory. The Expected Length field (described as Defined Length in the Test Log) and the Length field (described as Actual Length in the Test Log) show the number of bytes. They show any difference

between actual and expected field lengths. The binary content is formatted using MIME base64 and Hexadecimal presentation.

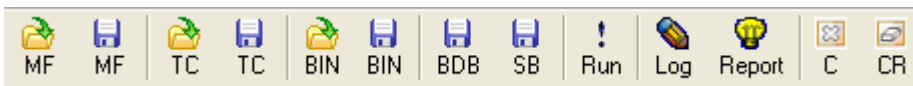


When a Manifest or Test Case is tested the “Name” (e.g., 000_Entire-fields-pass) and “Description” (Entire-fields-pass) are taken from the Manifest or Test Case. When a Binary file is tested against a Manifest, “Name” and “Description” are taken from the Manifest. “File Name” is the binary file name.

Help menu

Displays a Disclaimer and the Contributors’ list (Disclaimer option) and the CTS version (About option)

3.2.3 Icon Bar



The Icon bar (#3) provides the same functionality as the Menu bar (see Section 3.2.2), and also includes a “Run” icon. Click the Run Icon to run a test on the selected Manifest, Test Case or Binary file.

3.2.4 Name (Manifest/Test Case) and Description Panel

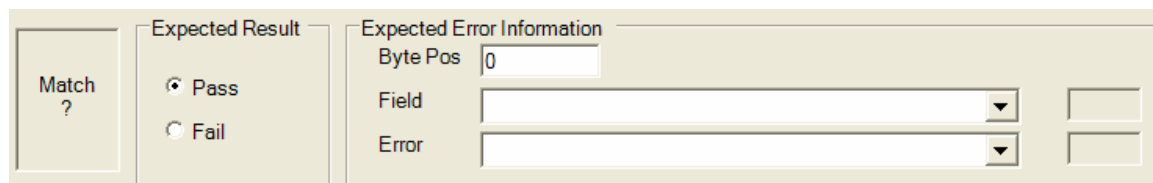


This Panel (#4) provides the following information:

a) Name: Displays and allows editing of the Manifest or Test Case name. This name is included in the Test Log and Test Report generated by testing the Manifest or Test Case. Note: This may be different from the Manifest or Test Case file name.

b) Description: This text appears in the Test Log and Test Report. It is highly recommended that this text include high level information on the Manifest or Test Case (e.g., format, author, intended purpose such as “Pass”/”Fail”).

3.2.5 Expected Result Panel



This panel (#5) is used during testing of Test Cases. It also displays information when Manifests and Binary files are tested. The panel consists of the following elements:

a) Match panel: Indicates whether there is a match between the Expected Result (“Pass”/”Fail”) and the Actual Result (“Pass”/”Fail”). The window is highlighted in **green** when the results match and in **red** when they don’t.

b) Expected Result panel

- When an existing Test Case is loaded, the Expected Result panel indicates whether the Test Case is expected to “Pass” or to ”Fail”. See a discussion on Test Cases in Section 4.2.1 of the “Overview” document – “Test Case Tester”.
- When developing a Test Case, set the value of the expected result in this panel.
- The Expected Results for all Manifests should be “Pass”.

c) Expected Error Info:

- For a Manifest, the data should be as shown in the screenshot above (e.g., Byte Pos equal 0, Field and Error blank).
- For a Test Case that is expected to “Pass”, the Expected Error Information panel should also be as shown in the screenshot above.
- For a Test Case that is expected to “Fail”, this panel displays the exact reason why it is expected to fail.

It is critical to understand the Match Panel to develop Test Cases. As shown below, the Match panel shows the relationship between the Expected Result and the Actual Result when a test is run.

The Match panel is blank when the CTS starts or when the Edit window has been cleared.

The screenshot shows a panel with a 'Match ?' button on the left. To its right is the 'Expected Result' section with radio buttons for 'Pass' (selected) and 'Fail'. Further right is the 'Expected Error Information' section with a 'Byte Pos' input field containing '0', and two dropdown menus for 'Field' and 'Error', each with an empty text box to its right.

The CTS distribution includes a number of Test Cases. All of them “Pass”. Some of them “Pass” because the self contained binary data conform to Patron Format A. “Pass” is indicated by the color green and the word “Match”.

This screenshot is identical to the previous one, but the 'Match ?' button is now a solid green box with the word 'Match' written in white text.

The screenshot shows a 'Test' panel. On the left is a 'Run Test' button with a warning icon. To its right is the 'Actual Result' section, which displays 'PASS' in a green box. Further right are three input fields: 'errorByteArrayPos' with '0', 'errorField' with '_Global', and 'errorNum' with 'NO_ERROR'. An 'All Errors' button is located on the far right.

Other Test Cases “Pass” because although the self contained binary data does not conform to the Patron Format, the actual error(s) reported by the CTS matches the expected result (see below).

Match

Expected Result

Pass

Fail

Expected Error Information

Byte Pos 4

Field SBH_patron_format_owner (1) 1

Error Global: ERR_DUPLICATE_FIELD (108) 108

Test

Actual Result **FAIL** errorByteArrayPos 4 All Errors

errorField 1: SBH_patron_format_owner

errorNum 108: ERR_DUPLICATE_FIELD

When there is a mismatch between the Actual Result and the Expected Result the Matching Result will be “Fail”. “Fail” is indicated by the color **red** and the words “No Match”

No Match

Expected Result

Pass

Fail

Expected Error Information

Byte Pos 6

Field BDB_format_owner (3) 3

Error Global: ERR_UNKNOWN_VALUE (107) 107

Test

Actual Result **FAIL** errorByteArrayPos 4 All Errors

errorField 1: SBH_patron_format_owner

errorNum 108: ERR_DUPLICATE_FIELD

3.2.6 Fields and Binary Data Panel

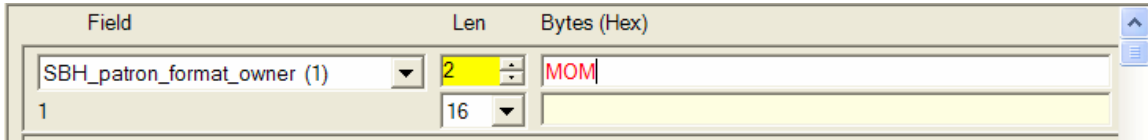
This panel (**#6**) lists the fields of a Manifest or Test Case, and the binary data associated with each field.

Field	Len	Bytes (Hex)
SBH_patron_format_owner (1)	2	00 1b
1	16	0 1B
SBH_patron_format_type (2)	2	00 01
2	16	0 1
BDB_format_owner (3)	2	00 01
3	16	0 1
BDB format type (4)	2	00 00
Fields	38	Total Length 97

a) Field: The ANSI INCITS 398-2008 PFA field names

b) Field number: Below each field is the field sequence number

c) Len: Field length in bytes. The Len Input box will be highlighted in yellow if the length is not the same as the number of data bytes.



d) Bytes (Hex): The binary data associated with each field in hexadecimal. If the input has a non-hex character, the text color changes to **red**.

e) Base and data box: The base dropdown box displays a user-selectable base (i.e., 2, 8, 10, 16, ASCII). The data box displays the binary data associated with the field in the selected base.

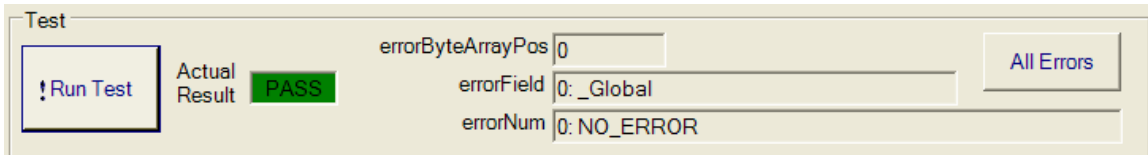
f) Fields: Displays the number of fields in the Manifest or Test Case (the number of “fields” includes the BDB and the SB).

g) Total length: Displays the sum of the length values of all the fields.

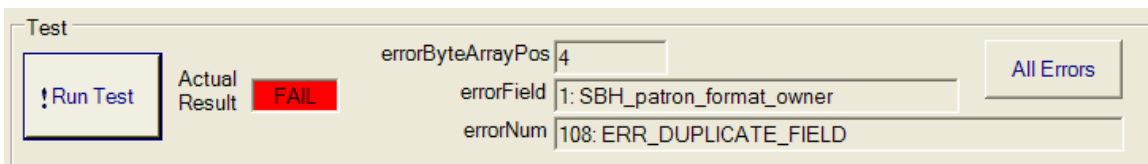
3.2.7 Test Panel

The Test Panel (**#7**) displays the results of the test last run.

When a Test Case or Manifest is tested and no errors are found the panel will display this information.



When errors are found, the first of any number of possible errors will be displayed. Clicking the “All Errors” button displays all the errors that were found during the test.



a) ! Run Test: Tests the Manifest, Test Case or Binary file in the window.

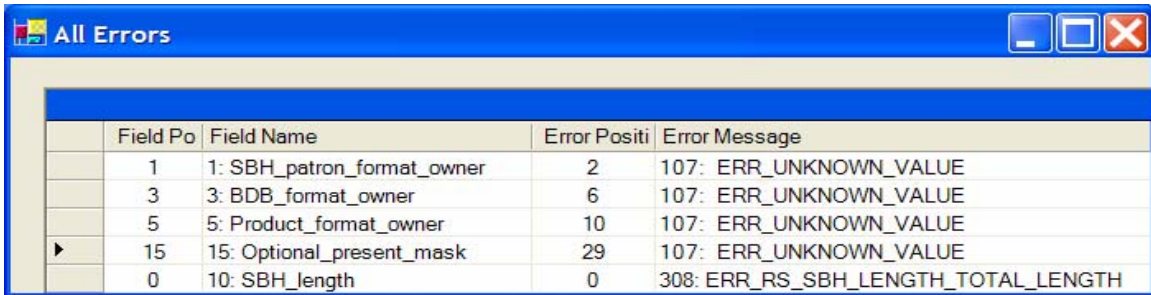
b) Actual Result: Indicates the test result as “Pass”/“Fail”. See also Section 3.2.5.

c) errorByteArrayPos: Indicates the error byte position where the error was found.

d) **errorField:** Indicates the Field in which the error was found.

e) **errorNum:** Indicates the Error number and its description.

f) **All Errors:** Displays all the errors that were found during this test and inserts all the errors in the Test Log and Test Report. See Annex A for error message descriptions.

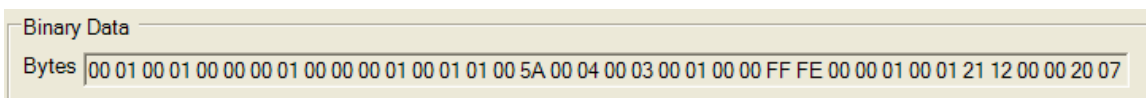


Field Po	Field Name	Error Positi	Error Message
1	1: SBH_patron_format_owner	2	107: ERR_UNKNOWN_VALUE
3	3: BDB_format_owner	6	107: ERR_UNKNOWN_VALUE
5	5: Product_format_owner	10	107: ERR_UNKNOWN_VALUE
15	15: Optional_present_mask	29	107: ERR_UNKNOWN_VALUE
0	10: SBH_length	0	308: ERR_RS_SBH_LENGTH_TOTAL_LENGTH

Annex B and C show screenshots of a Test Log and a Test Report that include a list of all the errors encountered by the CTS during a test. When “All Errors” are included in the Test Log and Test Report, they will not show the Test Result verdict on whether the Actual Result is the same than the Expected Result because a failure may have been caused by more than one error. If the test passed, the Test Log and Test Report will show a “Pass” message.

3.2.8 Binary Data Panel

This panel (#8) displays the concatenated binary data from the Fields and Binary Data panel. The bytes appear in fields sequence order.



3.2.9 Developing Manifests, Test Cases and Binary Files

3.2.9.1 Developing a Manifest

The CTS distribution includes a Manifest named “000_All_fields_pass.xml” that includes all the fields specified in Patron Format A. It is recommended to use this Manifest as the basis for developing other Manifests.

3.2.9.1.1 Making Minor Changes to an Existing Manifest

This clause gives step-by-step instructions for changing the BDB length in an existing Manifest. Similar steps are used to make other minor changes to a Manifest.

1. Open the CTS to the TestCase/Manifest/Binary File Editor window (see Section 3.2)
2. Open the Manifest "000_All_fields_pass.xml".
3. In the Test panel, click the "! Run Test" button and notice the Manifest passes.
4. Optionally, in the Name and Description panel, change the Name and/or Description.
5. In the Fields and Binary Data panel navigate to the field "BDB_length (11)". Using the "Bytes (Hex)" field enter a different BDB length such as "00 02".
6. Click the "! Run Test" button and notice the Manifest now fails.
7. Navigate to the field "Biometric_data_block (37)".
 - (a) Using the "Len" field set the length to value entered above in the "BDB_length (11)" field.
 - (b) Using the "Bytes (Hex)" field, enter bytes so that the number of bytes corresponds to the new length.
8. Click the "! Run Test" button and notice the Manifest now passes.
9. Optionally, save the Manifest.

3.2.9.1.2 Developing a Manifest from Scratch

1. Prepare a list of the fields the Manifest will contain. The list should include the field names, their length in bytes, and an example of valid binary data (in hexadecimal notation) for each field.
2. Open the CTS to the TestCase/Manifest/Binary File Editor window (see Section 3.2).
3. If the Editor window contains data, from the "Edit" menu select "Clear" to erase the window.
4. In the Name and Description panel, enter a suitable "Name" and "Description" for the Manifest.
5. In the Expected Result panel, ensure the Expected Result is "Pass".
6. In the Fields and Binary Data panel, set "Fields" to the number of fields the Manifest will contain.
7. In the Fields and Binary Data panel, for each field the Manifest is to contain:

- (a) Using the “Field” dropdown box, select the Field name for that position in the Manifest.
 - (b) Using the “Len” field, set the byte length of the field.
 - (c) Using the “Bytes (Hex)” field, enter the valid sample hexadecimal data for that field.
 - (d) To save or resave the Manifest at any time, from the “File” menu select “Save Manifest...”
8. When the fields have been entered it is recommended that the Manifest be saved.
 9. In the Test panel, click the “! Run Test” button. If an error occurs, correct the error in the corresponding field in the Fields and Binary Data panel, and click the “! Run Test” button again. Repeat this process until all errors have been corrected.
 10. Save the Manifest.

3.2.9.1.3 Developing a Manifest by Editing the XML Directly

A Manifest is an XML file and in some cases it is convenient to use an editor such as Notepad while developing a manifest. Examples include moving a misplaced field to its proper location and deleting an unneeded field. This clause gives step-by-step instructions for switching the order of the first two fields and duplicating the third field using Notepad on a copy of an existing Manifest.

1. In Windows Explorer, select the Manifest “000_All_fields_pass.xml”.
2. From the “Edit” menu select “Copy”.
3. From the “Edit” menu select “Paste”.
4. Using Notepad, open this new copy of the Manifest.
5. Change <NumberOfFields>38</NumberOfFields>
to <NumberOfFields>39</NumberOfFields>
6. Select this text:

```
<Field Sequence="1" Name="SBH_patron_format_owner" DefinedLength="2"
ActualLength="2" BytesBase64="ABs=" Hex="001B" />
```
7. From the “Edit” menu select “Cut”.

8. Insert a blank line after this line:

```
<Field Sequence="2" Name="SBH_patron_format_type" DefinedLength="2"
ActualLength="2" BytesBase64="AAE=" Hex="0001" />
```
9. Position the cursor on this blank line, then from the “Edit” menu select “Paste”.
10. Use similar editing commands to duplicate the third field. The first four fields should now be like this:

```
<Field Sequence="2" Name="SBH_patron_format_type" DefinedLength="2"
ActualLength="2" BytesBase64="AAE=" Hex="0001" />
<Field Sequence="1" Name="SBH_patron_format_owner" DefinedLength="2"
ActualLength="2" BytesBase64="ABs=" Hex="001B" />
<Field Sequence="3" Name="BDB_format_owner" DefinedLength="2"
ActualLength="2" BytesBase64="AAE=" Hex="0001" />
<Field Sequence="3" Name="BDB_format_owner" DefinedLength="2"
ActualLength="2" BytesBase64="AAE=" Hex="0001" />
```
11. Update the “Sequence” values.

```
<Field Sequence="1" Name="SBH_patron_format_type" ...
<Field Sequence="2" Name="SBH_patron_format_owner" ...
<Field Sequence="3" Name="BDB_format_owner" ...
<Field Sequence="4" Name="BDB_format_owner" ...
<Field Sequence="5" Name="BDB_format_type" ...
<Field Sequence="6" Name="Product_format_owner" ...
<Field Sequence="7" Name="Product_format_type" ...
<Field Sequence="8" Name="Device_format_owner" ...
<Field Sequence="9" Name="Device_format_type" ...
<Field Sequence="10" Name="Record_attributes" ...
<Field Sequence="11" Name="SBH_length" ...
...
```
12. From the “File” menu select “Save”.
13. Open the CTS to the TestCase/Manifest/Binary File Editor window (see Section 3.2 and the window screenshot).
14. Open the Manifest created in the above steps and note the reversed and duplicated fields.

3.2.9.2 Developing a Binary File

Developers may use the CTS to easily develop sample binary files similar to the ones that will be generated by a product.

An effective way to develop a sample binary file is to start from a Manifest, make any needed changes to the Manifest’s binary data, and then save the changed binary

data to a binary file. The step-by-step instructions below describe how to do this. Note that this example produces an invalid binary file.

1. Open the CTS to the TestCase/Manifest/Binary File Editor window (see Section 3.2).
2. Open the Manifest named "000_All_fields_pass.xml".
3. Click the "! Run Test" button in the Test panel to test this Manifest with its binary data. Notice that the test passes, indicating that this is a valid Manifest.
4. In the Binary Data panel, note that the first two bytes displayed are "00 1B".
5. In the Fields and Binary Data panel, change the Bytes in Field 1, SBH_patron_format_owner (1), from "00 1B" to any other value such as "00 1C" or "FF FF".
6. In the Binary Data panel, note the first two bytes have changed to correspond to the changes made in the previous step.
7. Click the "! Run Test" button in the Test panel and notice the error information in the panel.
8. Optionally, from the "File" menu select "Save Binary File..." to save the binary file.

3.2.9.3 Testing and Developing Test Cases

A user may not need to develop Test Cases since the primary use of them is to test the CTS. However this CTS allows the user to either test existing Test Cases (provided in the distribution) or to develop their own.

This clause gives step-by-step instructions for testing an existing Test Case and for creating a Test Case that verifies that an incorrect value in the SBH Patron Format Owner field does not pass. The same procedure can be used to modify an existing Test Case or to create these Test Cases to test the CTS for a specific set of PFA fields.

1. Open the CTS to the TestCase/Manifest/Binary File Editor window (see Section 3.2).
2. Open the Test Case named "000_All_fields_pass.xml"
3. In the Test panel, click the "! Run Test" button to verify that this Test Case passes. This is the procedure to perform a test of an existing Test Case.
4. In the Name and Description panel, change the Name to, for example: "Field_01_Bad_PFO_Value_Fails".

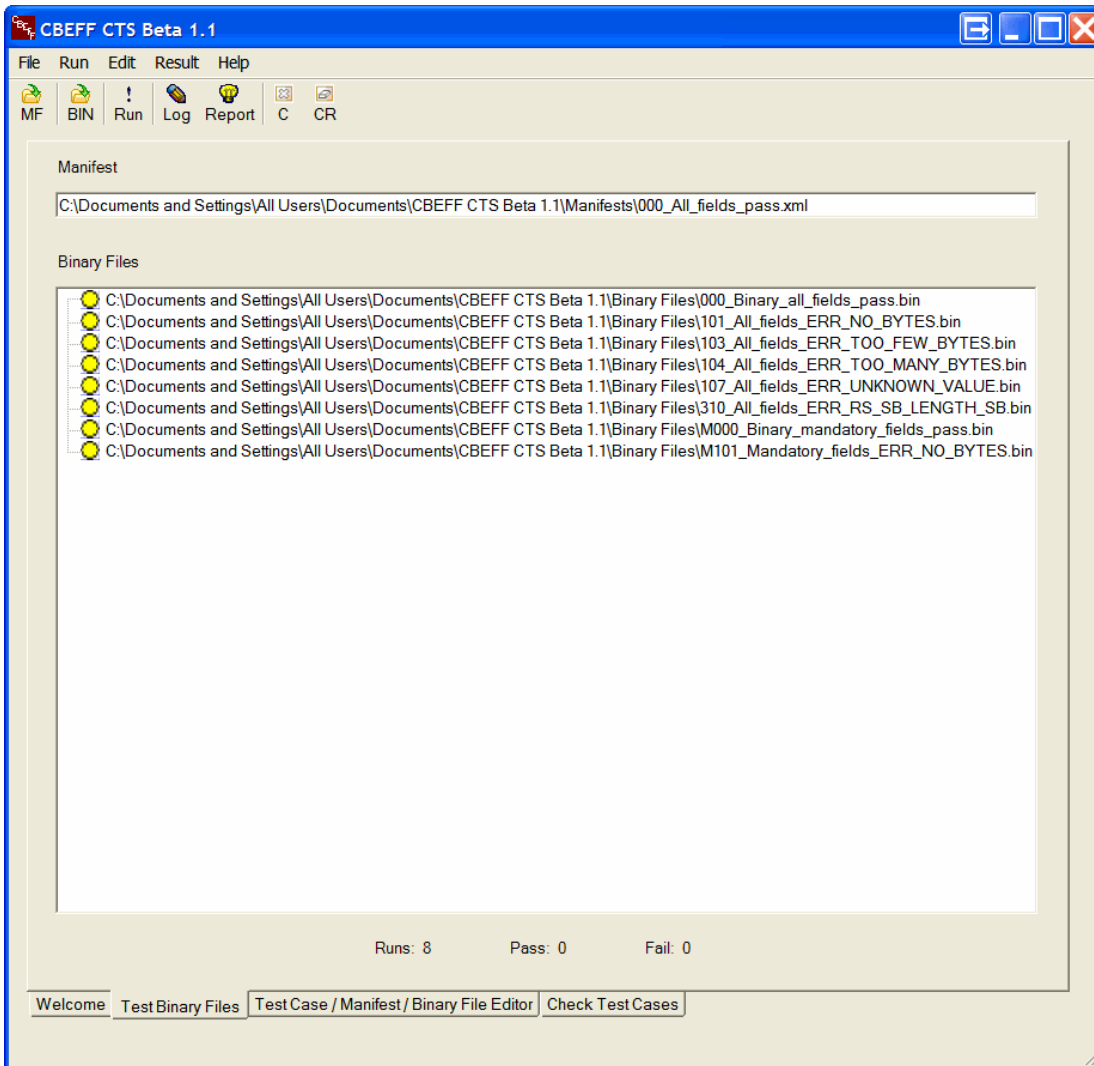
5. Change the Description to “Tests that have an invalid value for field 1, SBH Patron Format Owner do not pass.”
6. In the Expected Result panel, change the Expected Result to “Fail”.
7. In the Fields and Binary Data panel, change the Bytes in Field 1, SBH_patron_format_owner (1), from “00 1B” to any other value such as “00 1C” or “FF FF”.
8. In the Test panel click the “! Run Test” button and notice the error information in the panel. Note that although the Test Case is expected to “Fail”, the Match panel is highlighted in red. This is because the Expected Error Information does not match the Actual Results. See Section 3.2.5.
9. In the Expected Result panel, set the Expected Error Information to the same values displayed in the “Test” panel:
 - (a) Enter into “Byte Pos” the value displayed in “errorByteArrayPos”.
 - (b) Using the “Field” dropdown box, select the Field name displayed in “errorField”.
 - (c) Using the “Error” dropdown box, select the error number displayed in “errorNum”.
10. Click the “! Run Test” button. Note that the Match panel in the Expected Result panel indicates success since the Expected Results are the same as the Actual Results. The Test Case is complete, as the CTS verifies that an incorrect SBH Patron Format Owner does not pass.
11. Optionally, save the Test Case.

3.2.9.4 Testing Binary files

Binary files are tested in the “Test Binary File” window. This window is read-only; binary files are not changed.

3.3 Test Binary Files Window

See a screenshot of the window in the following page.



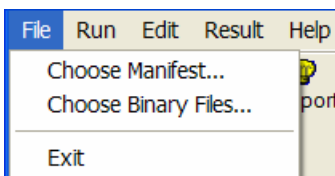
3.3.1 Title Bar

The title bar displays the CTS name and version number.

3.3.2 Menu Bar

Includes the following options: File, Run, Edit, Result and Help.

File menu



These options allow the operator to choose a Manifest or a Binary file or close the CTS.

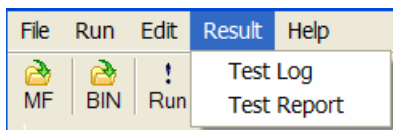
Run menu:

Select this option to test Binary files against a Manifest.

Edit menu

Select this option to clear the window.

Result menu



The Result menu is used to display the Test Log and/or the Test Report. The binary file names are included in the Test Log and Report. Examples of a Test Log and a Test Report are shown in Section 3.2.2. These Logs and Reports have the same format for both Manifest and Test Case testing.

Test Logs and Test Reports generated during Binary file testing also have the same formats as above with the exceptions of “TestSuite Type” which is “Binary” and the “File Name” which is the name of the binary file tested.

Help menu

Displays a Disclaimer and the Contributors’ list (Disclaimer option) and the CTS version (About option)

3.3.3 Icon bar

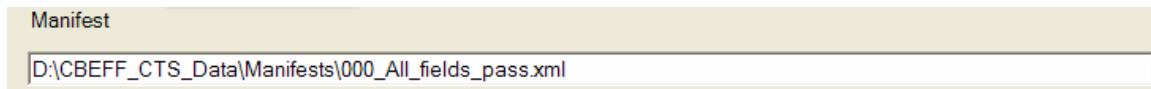


- **Manifest icon:** Click to select an existing Manifest.
- **Binary file icon:** Click to select one or more Binary files.
- **! Run icon:** Click to start the test of the selected Binary files against the selected Manifest.
- **Log icon:** Click to display the Test Log.
- **Reports icon:** Click to display the Test Report.

- **C Icon:** Click to clear the window.
- **CR Icon:** Click to clear the results.

3.3.4 Manifest Open panel

This panel displays the name of the Manifest that will be used to test a binary file or set of binary files.

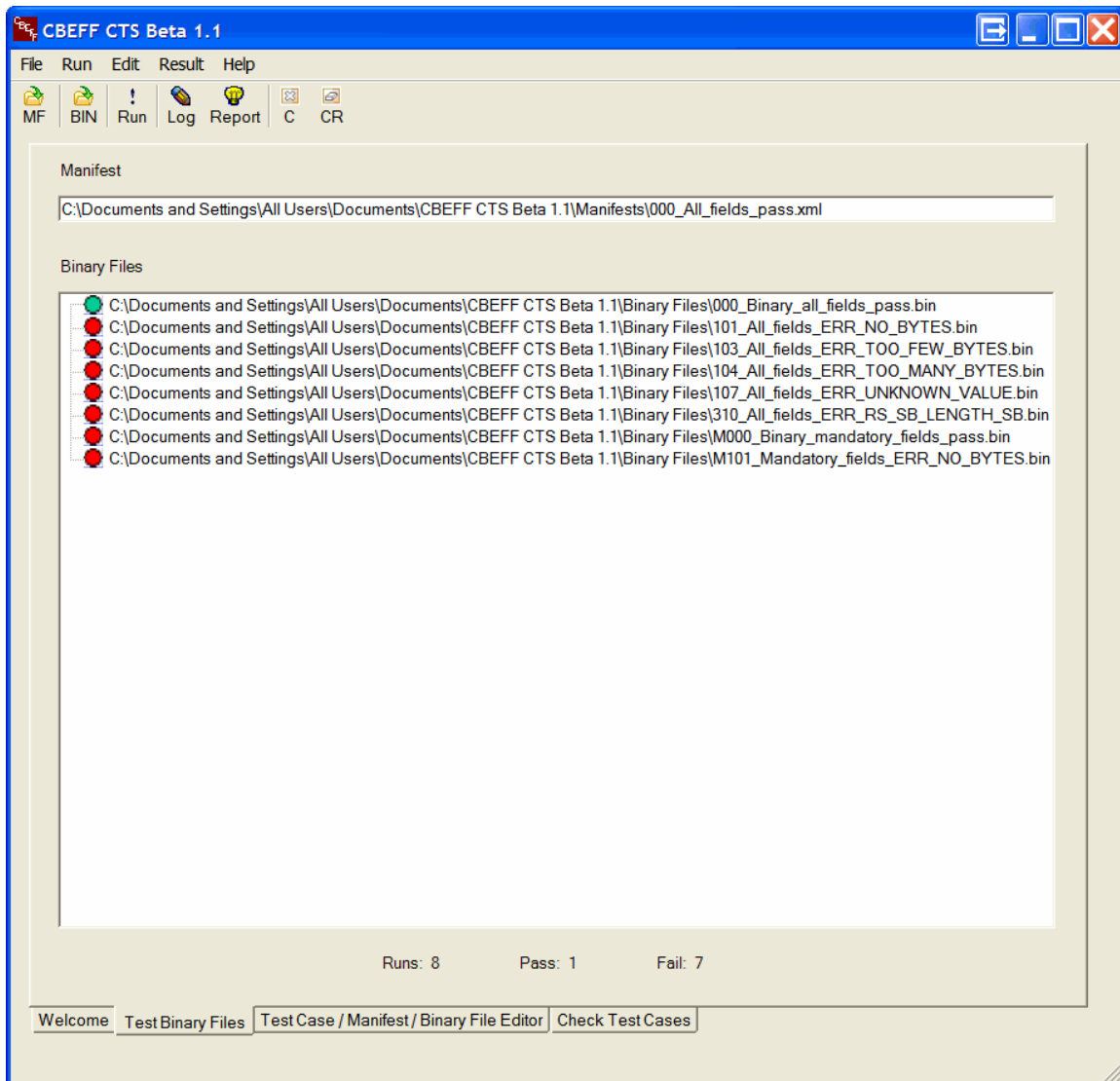


3.3.5 Testing Binary Files

1. Choose a Manifest. In normal use it is expected that this would be a “valid” Manifest for the binary files to be tested.
2. Choose a Binary file or a set of Binary files. The file name(s) will be displayed in the Binary Files window.
3. Run the test.

One of the following Status Indicators associated with each binary file selected will display one of the colors shown below:

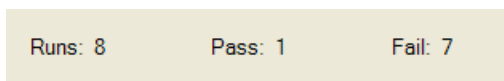
- Yellow: The result is undecided.
- Green: The test passed.
- Red: The test failed.



4. Test results are written to a Test Log and a Test Report. The Test Log and Test Report include information on all the Test Cases concurrently tested.

Double-click a Binary File name to open the Binary File and Manifest in the Test Case/Manifest/Binary File Editor window (see Section 3.2).

3.3.6 Test Status Panel

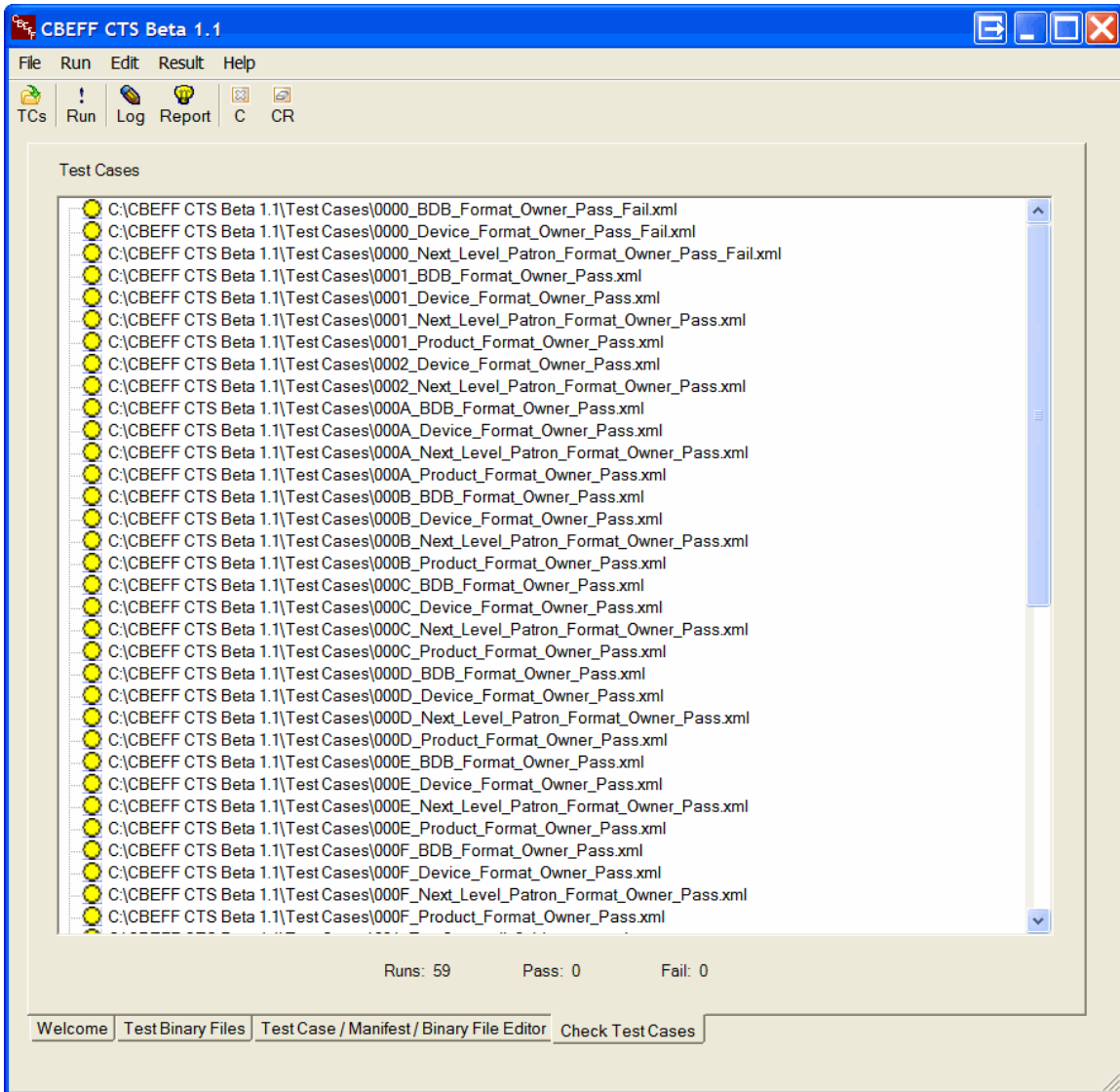


- **Numerical results:**

- **Runs:** Total number of tests run.

- **Pass:** the number of tests that passed.
- **Fail:** the number of tests that failed

3.4 Check Test Cases Window



This window is used to test Test Cases. A number of Test Cases are included in the CTS distribution. See Section 2.2.1 for sample data location.

3.4.1 Title Bar

The title bar displays the CTS name and version number.

3.4.2 Menu Bar

Includes the following options: File, Run, Edit, Result and Help.

File menu

These options allow the operator to choose one or more Test Cases or close the CTS.

Run menu:

Select this option to test the selected Test Cases.

Edit menu

Select this option to clear the window.

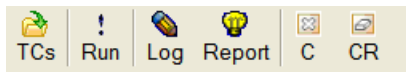
Result menu

The Result menu is used to display the Test Log and the Test Report (see Section 3.2.2).

Help menu

Displays a Disclaimer and the Contributors' list (Disclaimer option) and the CTS version (About option)

3.4.3 Icon Bar



- **TCs Icon:** Click to open a Test Case or a set of Test Cases.
- **Run Icon:** Click to test the selected Test Cases.
- **Log Icon:** Click to view the Test Log.
- **Report Icon:** Click to view the Test Report.

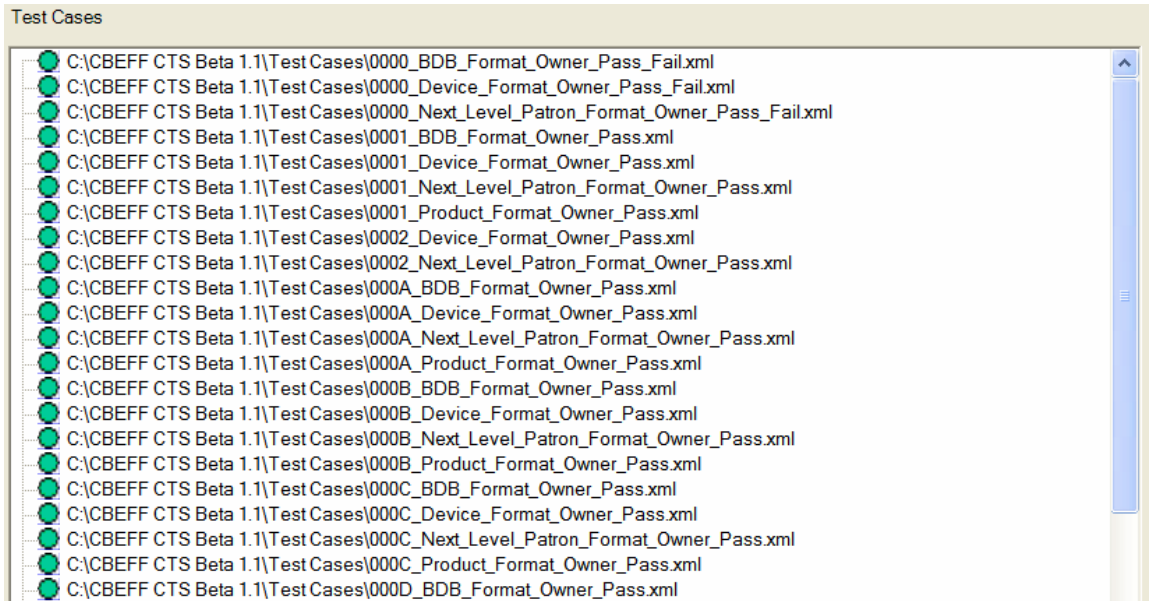
The Test Log and Test Report include information on all the Test Cases concurrently tested.

- **C Icon:** Click to clear the window.
- **CR Icon:** Click to clear the results.

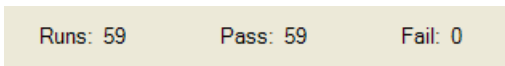
3.4.4 Test Case Panel

Indicates whether the actual results match the expected results. The Status Indicators display one of these colors:

- Yellow: The result is undecided.
- Green: The test passed.
- Red: The test failed.



3.4.5 Test Status Panel



- **Numerical results:** see Section 3.3.6.

3.4.6 Test Case Analysis

Double-click a Test Case to open it in the Test Case/Manifest/Binary File Editor window.

Annex A. List of Error Messages

There are a number of errors that can be encountered while running a test. They are related to the following:

- a) errors found in Mandatory fields
- b) errors found in Conditional and Optional fields
- c) errors derived from an incorrect relationship between fields (e.g., Record Attributes – SBH Length value & SBH # of bytes)
- d) errors derived from other required missing relationships between fields (e.g., FO/FT, creator content subfield vs. creator length subfield missing)

Table A.1 – List of Error Messages

Num	Error Name	Code	Comment
1	ERR_NO_BYTES	101	Manifest, Test Case or Binary file testing: No bytes were found
2	ERR_UNDEFINED_FIELD	102	Manifest or Test Case testing: Undefined field name
3	ERR_TOO_FEW_BYTES	103	Manifest or Test Case testing: The field length is greater than the number of bytes. Binary File testing: The number of bytes in the binary file is less than the number of bytes specified in the manifest.
4	ERR_TOO_MANY_BYTES	104	Manifest or Test Case testing: The field length is less than the number of bytes. Binary File testing: The number of bytes in the binary file is greater than the number of bytes specified in the manifest.
5	ERR_TOO_FEW_FIELD_BYTES	105	Manifest or Test Case testing: The number of bytes in the field under test is less than the number of bytes specified in the standard for the field.
6	ERR_TOO_MANY_FIELD_BYTES	106	Manifest or Test Case testing: The number of bytes in the field under test is greater than the number of bytes specified in the patron format for the field.
7	ERR_UNKNOWN_VALUE	107	Manifest, Test Case or Binary file testing: An unknown or incorrect value was found in the field under test.
8	ERR_DUPLICATE_FIELD	108	Manifest or Test Case testing: A duplicate field has been found in a test case or manifest under test.
9	ERR_MANDATORY_FIELD_MISSING	109	Manifest or Test Case testing: A mandatory field was not found in the manifest or test case.
10	ERR_BDB_MISSING_IN_RECORD	110	Manifest, Test Case testing: The BDB is missing.

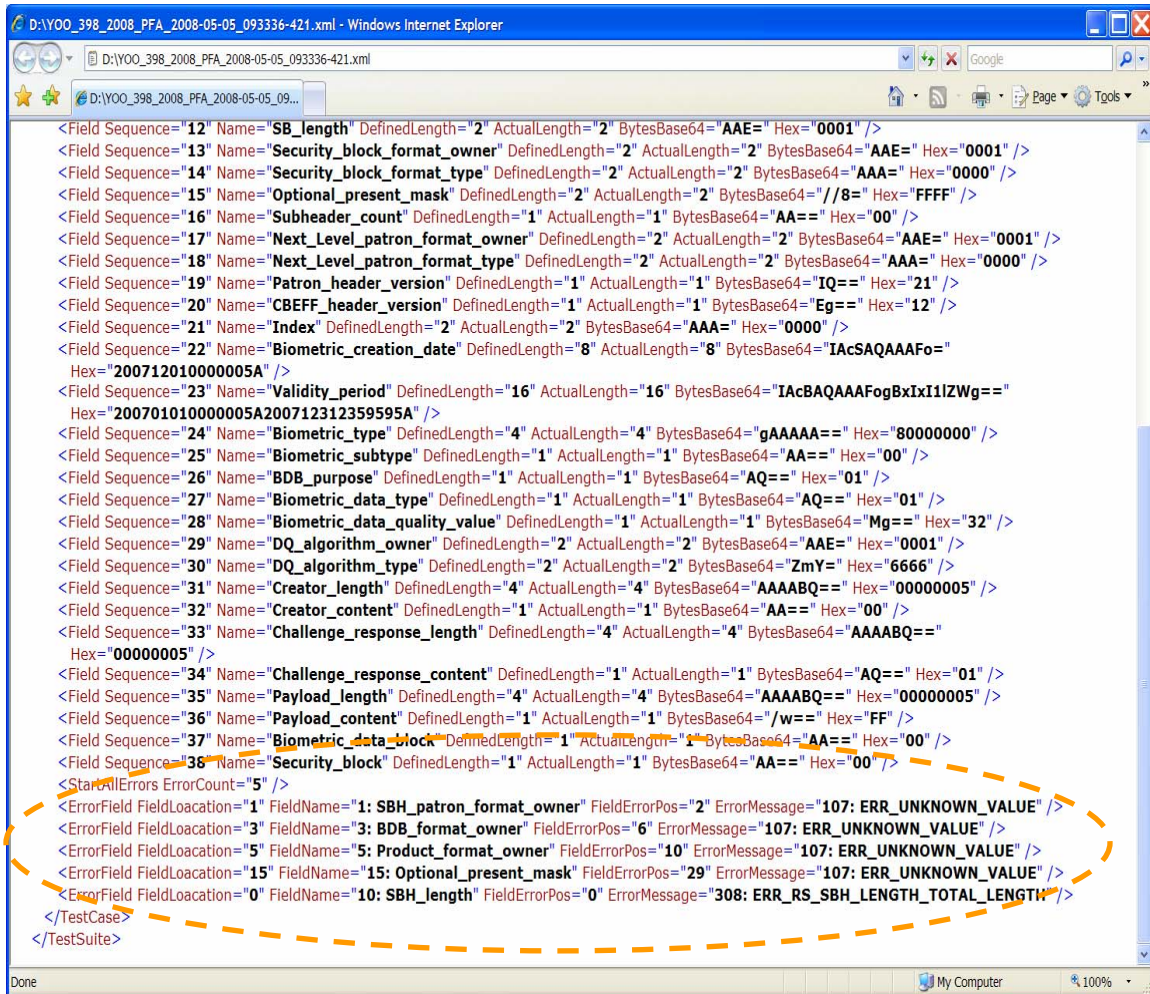
Num	Error Name	Code	Comment
11	ERR_SB_MISSING_IN_RECORD	111	Manifest or Test Case testing: The SB length field in the manifest or test case indicates that the SB must be present but it is missing.
12	ERR_FIELD_ORDER	112	Manifest or Test Case testing: Mandatory, conditional or optional fields are out of sequence.
13	ERR_PAIR_SBH_PATRON_FT_OWNER	201	Manifest or Test Case testing: SBH Patron Format Owner subfield is missing.
14	ERR_PAIR_SBH_PATRON_FT_TYPE	202	Manifest or Test Case testing: SBH Patron Format Type subfield is missing.
15	ERR_PAIR_BDB_FT_OWNER	203	Manifest or Test Case testing: The BDB Format Owner subfield is missing.
16	ERR_PAIR_BDB_FT_TYPE	204	Manifest or Test Case testing: The BDB Format Type subfield is missing.
17	ERR_PAIR_PRODUCT_FT_OWNER	205	Manifest or Test Case testing: The Product Format Owner subfield is missing.
18	ERR_PAIR_PRODUCT_FT_TYPE	206	Manifest or Test Case testing: The Product Format Type subfield is missing.
19	ERR_PAIR_DEVICE_FT_OWNER	207	Manifest or Test Case testing: The Device Format Owner subfield is missing.
20	ERR_PAIR_DEVICE_FT_TYPE	208	Manifest or Test Case testing: The Device Format Type subfield is missing.
21	ERR_PAIR_SB_FT_OWNER	209	Manifest or Test Case testing: The Security Block Format Owner subfield is missing.
22	ERR_PAIR_SB_FT_TYPE	210	Manifest or Test Case testing: The Security_Block _Format Type subfield is missing.
23	ERR_PAIR_NL_PATRON_FT_OWNER	211	Manifest or Test Case testing: The NL Patron Format Owner subfield is missing.
24	ERR_PAIR_NL_PATRON_FT_TYPE	212	Manifest or Test Case testing: The NL Patron Format Type subfield is missing.
25	ERR_PAIR_DQ_ALGORITHM_OWNER	213	Manifest or Test Case testing: The DQ algorithm Owner subfield is missing.
26	ERR_PAIR_DQ_ALGORITHM_TYPE	214	Manifest or Test Case testing: The DQ algorithm Type subfield is missing.
27	ERR_PAIR_CREATOR_LENGTH	215	Manifest or Test Case testing: The Creator Length subfield is missing.
28	ERR_PAIR_CREATOR_CONTENT	216	Manifest or Test Case testing: The Creator Content subfield is missing.

Num	Error Name	Code	Comment
29	ERR_PAIR_CR_LENGTH	217	Manifest or Test Case testing: The Challenge Response Length subfield is missing.
30	ERR_PAIR_CR_CONTENT	218	Manifest or Test Case testing: The Challenge Response Content subfield is missing.
31	ERR_PAIR_PAYLOAD_LENGTH	219	Manifest or Test Case testing: The Payload Length subfield is missing.
32	ERR_PAIR_PAYLOAD_CONTENT	220	Manifest or Test Case testing: The Payload _Content subfield is missing.
33	ERR_PAIR_BIOMETRIC_DQ_VALUE	221	Manifest or Test Case testing: The Biometric Data Quality Value subfield is missing.
34	ERR_PAIR_DQ_ALGORITHM_IDENTIFIER	222	Manifest or Test Case testing: The DQ Algorithm Owner and Type subfields are missing.
35	ERR_INVALID_OPT_FIELD_PRESENT_MASK	301	Manifest, Test Case or Binary file testing: An optional field that is present in the manifest, test case or binary file is not identified in the Optional Field Present Mask.
36	ERR_RS_PERIOD_FIR_SEC	302	Manifest, Test Case or Binary file testing: The Validity Period field date relationship (not before through not after) is incorrect.
37	N/A	303	N/A
38	ERR_RS_CREATOR_CONTENT_LENGTH	304	Manifest, Test Case or Binary file testing: There is a relationship conflict between the Creator length field value and the length of the Creator content field.
39	ERR_RS_CR_CONTENT_LENGTH	305	Manifest, Test Case or Binary file testing: There is a relationship conflict between the Challenge Response Length field value and the length of the Challenge_response content field.
40	ERR_RS_PAYLOAD_CONTENT_LENGTH	306	Manifest, Test Case or Binary file testing: There is a relationship conflict between the Payload Length field value and the length of the Payload Content field.
41	ERR_RS_OPTIONAL_FIELD_MISSING	307	Manifest or Test Case testing: An optional field marked present in the Optional Fields Present Mask field is missing.
42	ERR_RS_SBH_LENGTH_TOTAL_LENGTH	308	Manifest or Test Case testing: The length value in the SBH Length field is different than the total length of the SBH (based on the length of each SBH field indicated in the Manifest or Test Case). Binary file testing: See Note 1.

Num	Error Name	Code	Comment
43	ERR_RS_BDB_LENGTH_BDB	309	Manifest or Test Case testing: The length value in the BDB Length field is different than the length of the BDB (based on the BDB length indicated in the Manifest or Test Case). Binary file testing: See Note 1.
44	ERR_RS_SB_LENGTH_SB	310	Manifest or Test Case testing: The length value in the SB Length field is different than the length of the SB (based on the SB field length indicated in the Manifest or Test Case). Binary file testing: See Note 1.
45	ERR_RS_MISMATCH_WITH REC_ATT_MASK	311	Mismatch between the Record Attributes field mask and the SBH/BDB/SB Length.
46	ERR_RS_INCORRECT_FIELD_ PRESENT	312	Manifest or Test Case Testing: Incorrect relationship between the SB Length field value and presence/absence of the SB FO/FT and/or the SB.
47	ERR_RS_BDB_LENGTH_SUB_HEAD ER	313	Incorrect relationship between the BDB Length field value and the Subheader Count field value.

Note 1: Although supported by the CTS architecture, this version of the CTS is released only with a SBH PFA testing module. It does not include modules (under development) to test BDBs or SBs. Therefore, this CTS is not able to detect the errors in the boundary between the BDB and the SB.

Annex B. Test Log Displaying “All Errors” Information



```
<Field Sequence="12" Name="SB_length" DefinedLength="2" ActualLength="2" BytesBase64="AAE=" Hex="0001" />
<Field Sequence="13" Name="Security_block_format_owner" DefinedLength="2" ActualLength="2" BytesBase64="AAE=" Hex="0001" />
<Field Sequence="14" Name="Security_block_format_type" DefinedLength="2" ActualLength="2" BytesBase64="AAA=" Hex="0000" />
<Field Sequence="15" Name="Optional_present_mask" DefinedLength="2" ActualLength="2" BytesBase64="//8=" Hex="FFFF" />
<Field Sequence="16" Name="Subheader_count" DefinedLength="1" ActualLength="1" BytesBase64="AA=" Hex="00" />
<Field Sequence="17" Name="Next_Level_patron_format_owner" DefinedLength="2" ActualLength="2" BytesBase64="AAE=" Hex="0001" />
<Field Sequence="18" Name="Next_Level_patron_format_type" DefinedLength="2" ActualLength="2" BytesBase64="AAA=" Hex="0000" />
<Field Sequence="19" Name="Patron_header_version" DefinedLength="1" ActualLength="1" BytesBase64="IQ=" Hex="21" />
<Field Sequence="20" Name="CBEFF_header_version" DefinedLength="1" ActualLength="1" BytesBase64="Eg=" Hex="12" />
<Field Sequence="21" Name="Index" DefinedLength="2" ActualLength="2" BytesBase64="AAA=" Hex="0000" />
<Field Sequence="22" Name="Biometric_creation_date" DefinedLength="8" ActualLength="8" BytesBase64="IacSAQAAAFo="
Hex="200712010000005A" />
<Field Sequence="23" Name="Validity_period" DefinedLength="16" ActualLength="16" BytesBase64="IacBAQAAAFogBxIxI1ZWg=="
Hex="200701010000005A200712312359595A" />
<Field Sequence="24" Name="Biometric_type" DefinedLength="4" ActualLength="4" BytesBase64="gAAAA=" Hex="80000000" />
<Field Sequence="25" Name="Biometric_subtype" DefinedLength="1" ActualLength="1" BytesBase64="AA=" Hex="00" />
<Field Sequence="26" Name="BDB_purpose" DefinedLength="1" ActualLength="1" BytesBase64="AQ=" Hex="01" />
<Field Sequence="27" Name="Biometric_data_type" DefinedLength="1" ActualLength="1" BytesBase64="AQ=" Hex="01" />
<Field Sequence="28" Name="Biometric_data_quality_value" DefinedLength="1" ActualLength="1" BytesBase64="Mg=" Hex="32" />
<Field Sequence="29" Name="DQ_algorithm_owner" DefinedLength="2" ActualLength="2" BytesBase64="AAE=" Hex="0001" />
<Field Sequence="30" Name="DQ_algorithm_type" DefinedLength="2" ActualLength="2" BytesBase64="ZmY=" Hex="6666" />
<Field Sequence="31" Name="Creator_length" DefinedLength="4" ActualLength="4" BytesBase64="AAAABQ=" Hex="00000005" />
<Field Sequence="32" Name="Creator_content" DefinedLength="1" ActualLength="1" BytesBase64="AA=" Hex="00" />
<Field Sequence="33" Name="Challenge_response_length" DefinedLength="4" ActualLength="4" BytesBase64="AAAABQ="
Hex="00000005" />
<Field Sequence="34" Name="Challenge_response_content" DefinedLength="1" ActualLength="1" BytesBase64="AQ=" Hex="01" />
<Field Sequence="35" Name="Payload_length" DefinedLength="4" ActualLength="4" BytesBase64="AAAABQ=" Hex="00000005" />
<Field Sequence="36" Name="Payload_content" DefinedLength="1" ActualLength="1" BytesBase64="/w=" Hex="FF" />
<Field Sequence="37" Name="Biometric_data_block" DefinedLength="1" ActualLength="1" BytesBase64="AA=" Hex="00" />
<Field Sequence="38" Name="Security_block" DefinedLength="1" ActualLength="1" BytesBase64="AA=" Hex="00" />
<StartErrors ErrorCount="5" />
<ErrorField FieldLocation="1" FieldName="1: SBH_patron_format_owner" FieldErrorPos="2" ErrorMessage="107: ERR_UNKNOWN_VALUE" />
<ErrorField FieldLocation="3" FieldName="3: BDB_format_owner" FieldErrorPos="6" ErrorMessage="107: ERR_UNKNOWN_VALUE" />
<ErrorField FieldLocation="5" FieldName="5: Product_format_owner" FieldErrorPos="10" ErrorMessage="107: ERR_UNKNOWN_VALUE" />
<ErrorField FieldLocation="15" FieldName="15: Optional_present_mask" FieldErrorPos="29" ErrorMessage="107: ERR_UNKNOWN_VALUE" />
<ErrorField FieldLocation="0" FieldName="10: SBH_length" FieldErrorPos="0" ErrorMessage="308: ERR_RS_SBH_LENGTH_TOTAL_LENGTH" />
</TestSuite>
```

Annex C. Test Report Displaying “All Errors” Information

D:\YOO_398_2008_PFA_2008-05-05_093336-421.html - Windows Internet Explorer

D:\YOO_398_2008_PFA_2008-05-05_093336-421.html

D:\YOO_398_2008_PFA_2008-05-05_09...

19	Patron_header_version	1	1	IQ==	21
20	CBEFF_header_version	1	1	Eg==	12
21	Index	2	2	AAA=	0000
22	Biometric_creation_date	8	8	IAcSAQAAAFo=	200712010000005A
23	Validity_period	16	16	IAcBAQAAAFogBxIxl1lZWg==	200701010000005A200712312359595A
24	Biometric_type	4	4	gAAAAA==	80000000
25	Biometric_subtype	1	1	AA==	00
26	BDB_purpose	1	1	AQ==	01
27	Biometric_data_type	1	1	AQ==	01
28	Biometric_data_quality_value	1	1	Mg==	32
29	DQ_algorithm_owner	2	2	AAE=	0001
30	DQ_algorithm_type	2	2	ZmY=	6666
31	Creator_length	4	4	AAAABQ==	00000005
32	Creator_content	1	1	AA==	00
33	Challenge_response_length	4	4	AAAABQ==	00000005
34	Challenge_response_content	1	1	AQ==	01
35	Payload_length	4	4	AAAABQ==	00000005
36	Payload_content	1	1	/w==	FF
37	Biometric_data_block	1	1	AA==	00
38	Security_block	1	1	AA==	00

All Errors Information

Field Location	Field Name	Error Pos.	Error Message
1	1: SBH_patron_format_owner	2	107: ERR_UNKNOWN_VALUE
3	3: BDB_format_owner	6	107: ERR_UNKNOWN_VALUE
5	5: Product_format_owner	10	107: ERR_UNKNOWN_VALUE
15	15: Optional_present_mask	29	107: ERR_UNKNOWN_VALUE
0	10: SBH_length	0	308: ERR_RS_SBH_LENGTH_TOTAL_LENGTH

My Computer 100%

Annex D. IBIA Identifiers

The IBIA identifiers for Patron Format Owner, BDB Format Owner, Product Owner, Device Owner, Security Block Format Owner and Quality Algorithm Owner are in the text file:

ValidBdbFormatOwners.txt

It is in the CTS installation folder. The default folder is

C:\Program Files\CBEFF CTS Beta 1.1\ValidBdbFormatOwners.txt

See: http://www.ibia.org/cbeff_new/cbeff_organizations.asp

To add an IBIA identifier, modify the file in a text editor such as Notepad. The required format is documented in the file. (Note that this is not an IBIA registration procedure.)