BioCTS: AN MRT
User Guide

Version 1.0

NIST/ITL CSD Biometric Conformance Test Software using

ANSI/NIST-ITL Machine Readable Tables

April 2017

Christofer J. McGinnis
ID Technology Partners (NIST Associate) Software Developer / Biometric Engineer

Dylan Yaga
NIST/ITL CSD Software Designer / Project Manager

National Institute of Standards and Technology (NIST)

Information Technology Laboratory (ITL)

Computer Security Division (CSD)

Contents

Tables and Figures

Disclaimer

NIST/ITL CSD BioCTS MRT Edition

October 2010

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 Sates 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.

Version History

Version	Revision Date	Authors	Description
1.0	2017-10-04	McGinnis, Christofer; Yaga, Dylan	Initial Version

Introduction

NIST/ITL CSD has supported the development of biometric conformance testing software for many years, including the Biometric Conformance Test Software (BioCTS) family of applications. BioCTS, which has been freely available[1] since 2012, supports biometric data interchange format standards developed by the International Committee for Information Technology Standards (INCITS), the International Organization for Standardization (ISO), and the American National Standard Institute (ANSI).

While BioCTS broadly supports many biometric specifications, versions that support the ANSI/NIST-ITL (AN-ITL) line of standards have received a great deal of development support. The evolving nature of the AN-ITL standard, its many profiles for specific application domains, and its extensive use in U.S. government, as well as internationally, all necessitate efforts to provide conformance testing solutions.

In early 2017, a suite of BioCTS applications was released to support user-defined requirements and profiles for AN-ITL specifications. These applications make use of configuration files to dynamically create parsing rules and conformance requirements for any version or profile of the AN-ITL standard. The configuration files utilize a format called ANSI/NIST-ITL Machine Readable Tables (MRTs), which are XML-based. The BioCTS applications making use of the MRTs are collectively referred to as BioCTS AN MRT. This document provides an overview of the features common among all BioCTS AN MRT applications, specific details about installing and operating each application, and details regarding the use of MRT configurations.

Motivation for Configurable Requirements

Development of BioCTS applications traditionally followed the publication of Conformance Testing Methodology (CTM) documentation, which specified the assertions required to sufficiently test for conformance to the requirements found in the related biometric standard. Based upon the assertions specified in those CTM documents, manual software development was required to code each of the assertions needed to test the requirements found in the standard.

This process requires a large amount of development time after the publication of the standard and related CTM, often leading to long delays in the release of conformance tools. This approach also defines conformance tests statically, meaning that (a) end users with domain-specific requirements and user-defined fields are not able to modify the conformance tests or parsing rules and (b) any modification to the base standard requirements or subsequent revision requires a new release of BioCTS applications.

To alleviate these issues, BioCTS must allow modification of test assertions and parsing rules, at least to some degree. A configuration file specifying this information would allow the software to respond to the needs of the end user. While the BioCTS team had previously considered developing a user configurable format file to support this functionality, the AN-ITL MRTs were already under development. The MRTs

---

[1] BioCTS is hosted by NIST and available for download at: https://www.nist.gov/itl/csd/biometrics/biometric-conformance-test-software-biocts

were chosen as the BioCTS configuration format due to the large number of standards and profiles already defined in the MRT format.

ANSI/NIST-ITL Machine Readable Tables

A machine-readable specification of the AN-ITL 1-2011 data format was created and published on NIST's website[2] in 2013. This specification defines MRTs that specify many aspects an AN-ITL biometric transaction including records, fields, and related data constraints. The MRTs are documented in an XML file format which software can easily accept, parse, and use while remaining human-readable.

Using MRTs, BioCTS AN MRT applications can accept an XML configuration file as part of the input to the program and can parse and interpret the XML file, generating tests dynamically. Any adjustments to the MRTs can be performed using a simple text editor, allowing users to define or modify transaction elements and requirements, define MRT files for new standards and applications, or remove requirements upon the release of errata or error corrections without waiting on a release of new BioCTS versions.

BioCTS MRT processes and compiles the data from individual MRT documents to produce testing processes for an AN-ITL transaction. Processing and compilation of the MRT documents is necessary because they are designed to be layered onto previous versions. For example, the AN-ITL 2013 update may be produced by layering three MRT files: AN-2011, AN-2013 error corrections, and AN-2013 new additions. After the MRT data is compiled, the test transactions are parsed and tested using the definitions found in the MRT files.

---

[2]ANSI/NIST-ITL Machine Readable Tables: https://www.nist.gov/itl/iad/image-group/ansinist-itl-standard-references

BioCTS AN MRT Applications

The BioCTS AN MRT applications are built upon the same foundational code libraries, meaning that there is significant overlap in their operation and output. This section details some of the common features among all BioCTS AN MRT applications.

Software Requirements

Supported Microsoft® Operating Systems

- Windows® 10 ™
- Windows® 8.1 ™
- Windows® 7 ™ Service Pack 1

BioCTS AN MRT applications target the Microsoft® .NET 4.5 Framework. Downloads can be found for version 4.5.2:

- Web Installer (https://www.microsoft.com/en-us/download/details.aspx?id=42643)
- Offline Installer(https://www.microsoft.com/en-us/download/details.aspx?id=42642)
- Links working as of 2017-04-10

Download and Installation

To download BioCTS MRT Edition, go to https://www.nist.gov/itl/csd/biometrics/biocts-machine-readable-tables. Navigate to the Software Downloads section, and download the latest release.

Once the download has finished, you may verify its authenticity using a SHA256 hash:

- Open PowerShell (Start>All Programs>Accessories>Windows PowerShell>Windows PowerShell)
- Change Directories to the download location, ("C:\Users\User\Downloads" in this example), type:
  `cd C:\Users\User\Downloads`
- Here we will use the **`cmdlet`** called **`Get-FileHash`**, passing it the file name, and telling it to use the SHA256 algorithm, type:
  `Get-FileHash .\FileName.zip -A SHA256`
- The `cmdlet` will get the file hash of the zip file and display it in PowerShell.
- The resulting file hash should match the file hash posted on the website exactly. If it does not, then there was a problem with the download. Please try again.

Using the compression software of your choice, decompress the ZIP file to a location for use.

MRT Configuration Processing

All BioCTS AN MRT software makes use of a core set of libraries that read, parse, and validate AN MRT configurations and files, known collectively as the **AN MRT Toolkit**. The formats of the configuration file and MRT files are described by documentation and XML Schema provided with the MRT files. BioCTS AN MRT software will first parse, validate, and link all AN MRT files prior to building conformance test methods and running conformance tests. See the section on Default MRTs for an overview of the MRT files included with BioCTS AN MRT applications and the standards they represent. The results of the AN MRT processing and validation is output to the location specified by the user or the default location for the application. The format of these results will be:

- A Summary Log providing an overview of the MRT file contents and linking relationships required to build a full configuration. MRT files are stackable, meaning that the final MRT configuration may rely on previous configurations, which are called parents. This relationship will be shown in the summary log.
- An Error Log if any errors are detected in the MRT configuration file or individual MRT files.

The user will be asked to correct MRT errors before proceeding, but some applications may allow the user to override this request and proceed to conformance testing.

Testing Overview

BioCTS generates a set of test results based on the contents of the specified MRT files. Some test results are static, meaning that they are required for any version of AN-ITL, and may not be modified or removed (see the section on Static (Non-Modifiable) Requirements). End users are able to create, modify, or remove MRT-specified requirements to meet the needs of specific applications and domains by modifying the XML MRT files. Pre-defined MRT configurations are freely available along with BioCTS AN MRT to provide support for specific versions of AN-ITL standards and profiles.

BioCTS does not make explicit conformance claims, because there is no guarantee that any set of test assertions is comprehensive.

Conformance Testing Support

All BioCTS AN MRT applications support L1 conformance testing for Traditional and XML encodings of AN-ITL transactions for all Record Types, Fields, and other entities as defined by the provided MRT files. The MRT files provided with the applications includes requirements for all entities defined by their respective base standards and profiles. For more information about the default MRT files, see the section on Default MRTs.

Testing Support for MRT Elements

The initial version of BioCTS AN MRT applications are capable of reading, parsing, and validating all MRT elements, but not all of the MRT elements are used to generate conformance tests. All elements related to L1 testing are supported, but some L2-related MRT elements are not supported. The following is a list of MRT elements not used in Version 1.0.0 of the BioCTS AN MRT applications to generate tests:

- DependPresenceReq
- DependPresenceOpt
- DependAbsence
- DependValue
- DependValueReq
- DependValueOpt
- DependOther
- InterFieldOccur
- InterFieldCount
- InterFieldValue
- InterFieldOther
- Exception

Any requirements contained in XMLPath that are not identified with the **AppliesTo{}** notation (See the section on <u>One XML Element Representing Many Traditional Entities</u>) are also not used for conformance test generation in the initial version of BioCTS AN MRT applications. L2 Conformance support for the above listed elements is expected to be added in a future release of BioCTS AN MRT applications.

Test Results
The result of any test is composed of at least four elements:

(1) Test Name
A string to define the test. Test names format depends upon the type of entity tested.

Entity	Format	Examples
Fields and Information Items (but not Subfields)	**[FieldID][FieldRef]-Description** **FieldID** and **FieldRef** are taken from the FieldDef.	<u>Field 1.001-LEN Min Subfield Count:</u> [01.001:T][1/LEN:T]-MinOccur <u>Info Item 1.001-FRC Data Type:</u> [01.003-A][1/CNT/FRC]-DataType
SubFields	**[FieldID][FieldRef]{Occurrence}-Description** This is the same format as Fields and Information Items, except that the SubField **Occurrence** is entered in braces.	<u>2nd Subfield in 1.003-CNT Info Item Count:</u> [01.003:T][1/CNT:T]{2}-InfoItemCount
Records	Record[**RecordType**]-**Description** Where the **RecordType** is the number of the Record Type being tested.	<u>Record Type-1 Field 1 is first:</u> Record[1]-Field001IsFirst
Transactions	Transaction-**Description**	<u>Transaction has data present:</u> Transaction-Data-Present
Description indicates the type of test being performed, and may or may not be the name of a specific element in the MRT documents.		

<p align="center">Figure 1 – Test Name Formats</p>

(2) Test Level
There are four valid levels of testing: L1, L2, L3, and Parse.

The level of testing is an indicator of the complexity and type of test being performed. Levels 1 through 3 (L1, L2, and L3) are described in the base standard, but roughly indicate the level of complexity required for the test, with 1 being the least complex and 3 the most complex. The MRT format does not currently support L3 testing. Many of the requirements defined by the MRTs are L1 (for example value,

character-type, occurrence, and data length tests). L2 testing includes inter-field and conditional testing. Many L2 tests have specific MRT XML elements to assist in their definition[3].

In addition to L1 through L3 tests, some tests are labeled as Parse. All Parse-level tests are defined in this document as static (non-modifiable) requirements. These tests are necessary for reliable parsing of any AN-ITL formatted transaction and cannot be modified using the MRTs.

(3) Test Result
There are four valid types of Test Result:

- Ok – The test passed
- Warning – The test passed, but warranted a warning statement (i.e. an "Unspecified" value, etc.)
- Error – The test failed
- Critical Error – The test failed and may be preventing other tests from being performed

(4) Test Message
A description of how the test was performed. The message sometimes includes an error message.

(Optional) Reference:

The test result may also include one or more optional references that assist in linking the tests back to their source in the MRT documents.

References take one of two forms:

Reference Type	Format
Explanation	**[Explanation: `Description`]** The `Description` may be used to explain why the test references a certain MRT element, or how the MRTs specify that the test is required.
Paths	**[Document Path: `FilePath`][Element Path: `XPath`] at Line Number `LN`** The `FilePath` is used to indicate the file path to the specific MRT document that contains the referenced MRT element that defines the test. The `Element Path` is used to indicate a path to the specific MRT element within the referenced document, and is in XPath notation. `LN` is the line number of the specified MRT element in the referenced document.

Figure 2 – Test Message Reference Formats

---

[3] Note: the initial versions of BioCTS AN MRT provide complete support for L1 MRT definitions, but only limited support for L2 MRT features. The initial version does support EBTS-specific TOT tests, which are L2.

The following is an example of a test result printed to a text file. The result includes two optional references, which describe the location of the MRT reference document and element as well as an explanation as to why those references are used.

```
--------------------------------------------------------------------------------------
 4.|      Test Name| [01.001:T][1/LEN:T]-MinOccur
--------------------------------------------------------------------------------------
   |          Level| L1
   --------------------------------------------------------------------------------------
   |         Result| Ok
   --------------------------------------------------------------------------------------
   |   Test Message| The the minimum number of subfields allowed is 1. The number of subfields
   |               | present is 1.
   --------------------------------------------------------------------------------------
   |      Reference| [Explanation: The <MinOccur> value is used. Note: a minimum of 1 is always
   |               | enforced because all tagged fields must have at least one Subfield.]
   --------------------------------------------------------------------------------------
   |      Reference| [Document Path: D:\biocts\trunk\Code\BioCTS_AN_MRT\BioCTS_AN_MRT\bin\Debug\confi
   |               | gurations\default\AN2011-FieldDefinition.xml]-[Element Path:
   |               | /FieldDefinitions/FieldDef[2]/MinOccur]
--------------------------------------------------------------------------------------
```

Figure 3 – Text Output Result Example

Test Reports

The test results will include text and XML logs unless turned off in the related application options. The logs will contain the results in the format outlined in this document. Results will also provide a Batch Result Summary if more than one file is tested. This summary will provide an overview of the testing, including which MRT configurations were used, the number of files passing and failing for each configuration, and a summary of total results by type for each file and configuration combination.

BioCTS AN MRT GUI

This section describes the BioCTS AN MRT GUI Edition V.1.0.0, which provides a powerful graphical interface for testing and editing AN-ITL transactions.

Running BioCTS MRT GUI Edition

The application may be launched by double clicking the BioCTS icon or executable.

Loading Screen

After launching the application, a BioCTS MRT Loading Window will appear. This loading screen will remain open until the initial configuration file[4] and MRT files are loaded and parsed. A status message indicator at the bottom of the screen will update to provide details about the loading process.
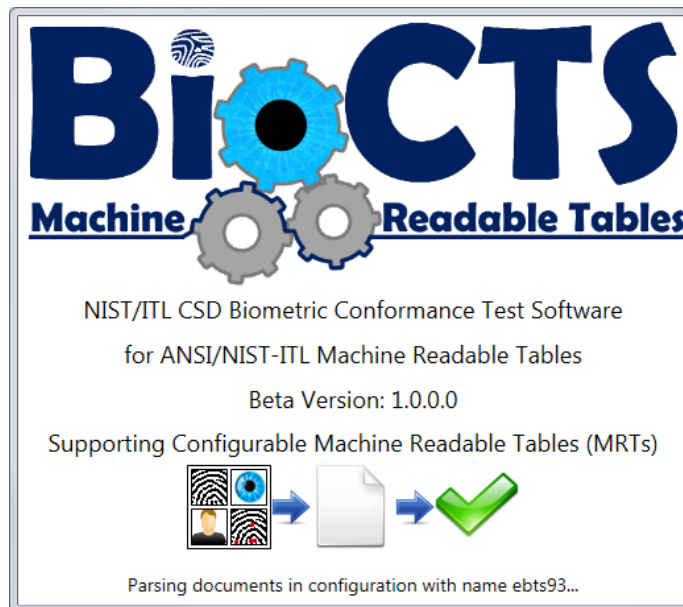

Figure 4 – Loading Screen

Batch Testing

This section outlines features of the user interface that assist the end user in loading, testing, and analyzing transactions. Any images of the software are provided for illustration only and may not represent the final software release.

The "Transaction Batch Test" tab allows multiple transactions (files) of any encoding to be tested in groups, and displays the overall results for each transaction in the "Files Under Test" pane. All MRT configurations loaded from the configuration file specified under the Options tab will be shown in the "Files Under Test" pane as column headers. Each loaded file path will appear as a new row. The

---

[4] The configuration file is set using the Options Tab. Upon the first time loading the application, the default configuration file provided with the application is loaded.

intersection of the configuration and file path will hold the result of the conformance test after it is completed.

Toolbar Buttons

Open Files: loads files into the Batch File List and removes any previously loaded files

Add Files: loads files into the Batch File List by adding them after any previously loaded files[5]

Test Files: tests all files loaded in the Batch File List

Stop Testing: stops all testing of the files in the Batch File List after the current file completes

Clear Results: clears all testing results from the Batch File List (but does not remove the files)

Clear Files: clears all files from the Batch File List (and removes the test results)

Find/Search: Opens text-search window for searching test results (also Ctrl + f). The F3 key locates the next matching result. The Esc key closes the Find/Search Window.

Results

The status banner at the bottom right side of the window displays messages regarding the tool's current actions. While testing, the banner displays "Testing Batch Files…".

---

[5] Drag and drop is supported on the File Under Test pane of the Batch tabs, so that transaction files can be dropped into the batch list. Files dropped into the batch list are added to the existing list of files. This feature also supports dropping a directory that includes transactions.
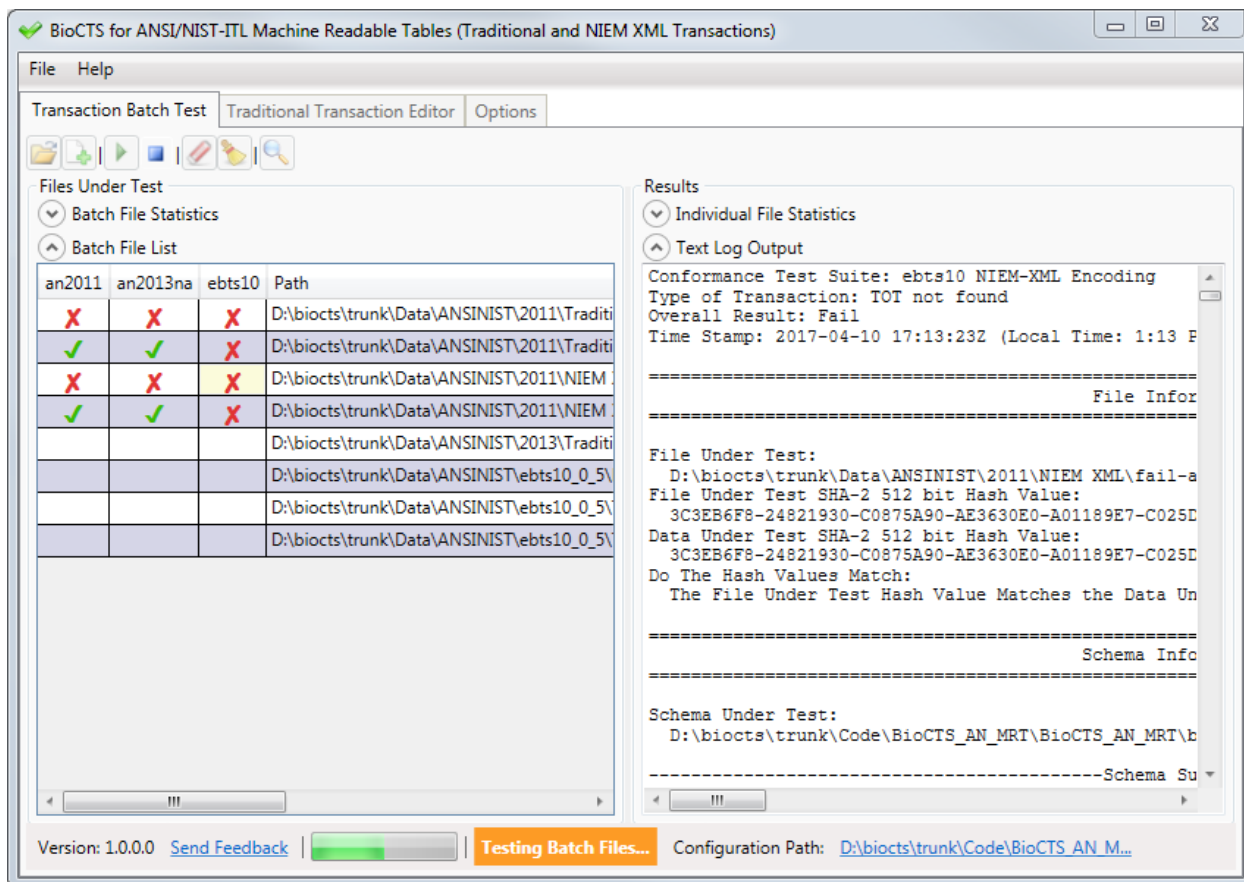
Figure 5 – Batch Testing In Progress

The "Batch File List" will display each transaction's overall result for each configuration, indicated by either:

- ✗ - Overall Result of Fail
- ✔ - Overall Result of Pass

Once all files in the Batch File List have completed testing, the status banner at the bottom right side of the window displays "Test Complete! …" as well as the testing time in green.

Textual output results for each transaction can be viewed by clicking on the desired result cell in the "File Under Test" pane. The complete textual results are displayed in the pane to the right.

Context Menu

A context menu is available when transaction files are loaded in the Batch File List found in the "File Under Test" pane. By right-clicking on any of the cells, a context menu is displayed with options to open the results files or the source data files. An option to open the transaction in the Traditional Transaction Editor is also available (Traditional encoding only).
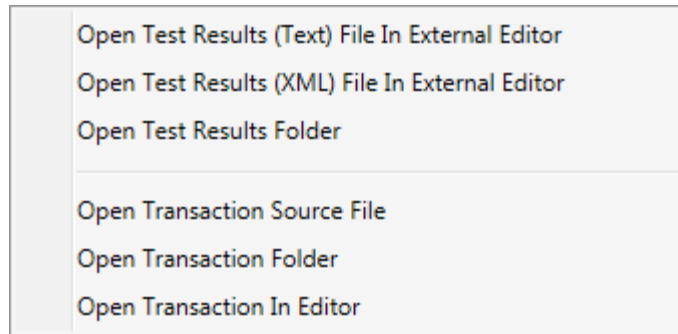
Figure 6 – Batch Result Context Menu

Stats

In addition to the detailed Textual Output Log, each file has a statistical breakdown of the results located just above the results pane. There is also a list of batch file stats just above the Batch File List that updates in real time. The batch stats indicate the overall files loaded, passing, and failing by encoding. Both of these stat windows are hidden by default, but may be expanded as shown below.

Figure 7 – Batch Testing Complete with Statistics

Transaction Editor

The Traditional Transaction Editor is capable of interactively displaying all entities contained in a traditionally encoded transaction, allowing a user to edit, add and remove records and their contents.
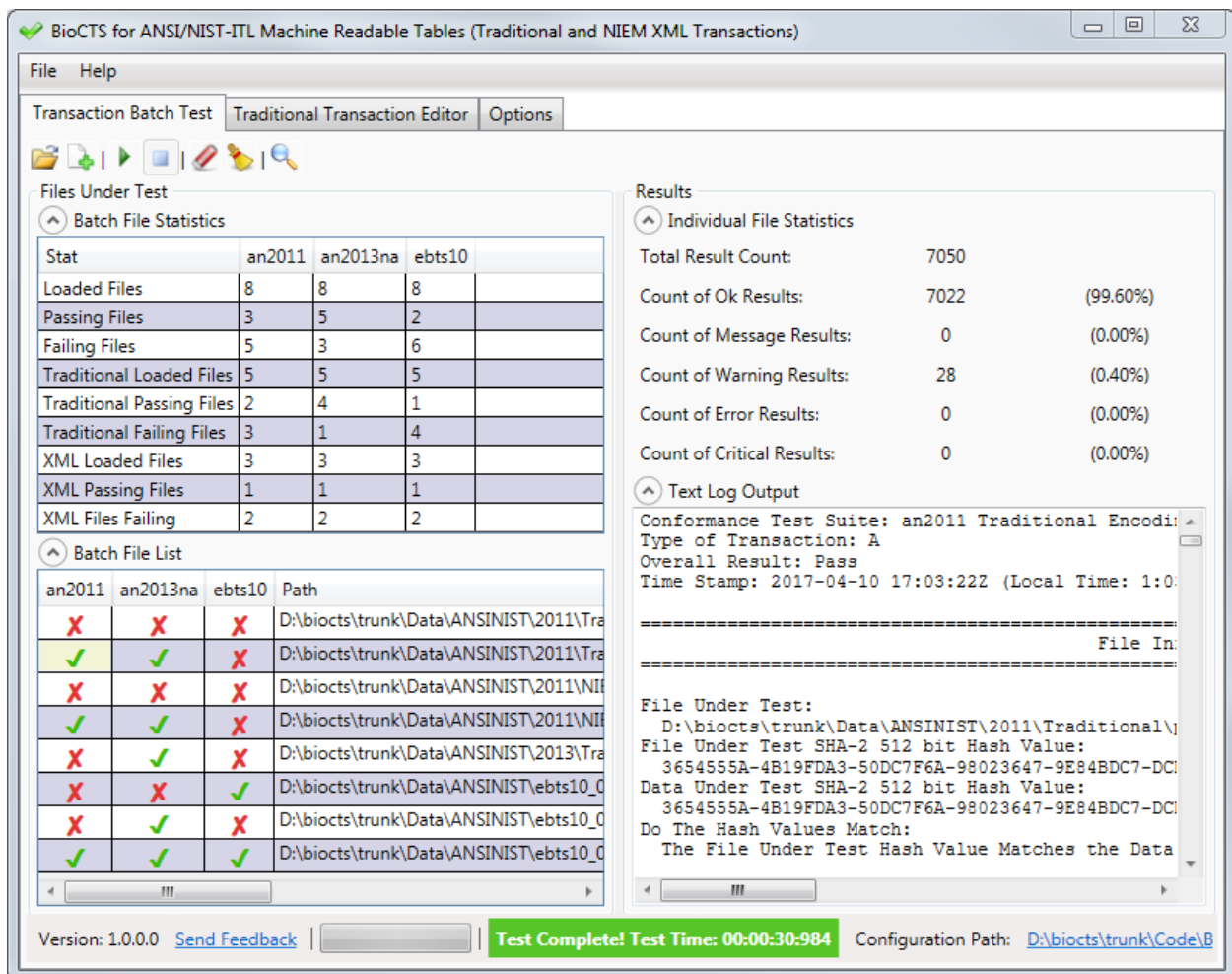
File Information

As files are loaded and edited, the File Information will update to reflect the changes. The information includes the file path, file size, file hash value, and the current modified data hash value.

Status Label

The overall testing results are shown at the bottom of the screen using a status banner. The banner will also display updates during file parsing and testing.

Configuration Selection

There is a "Selected Configuration" drop down selector at the top of the window, which allows the configuration to be changed (and the file to be automatically retested).
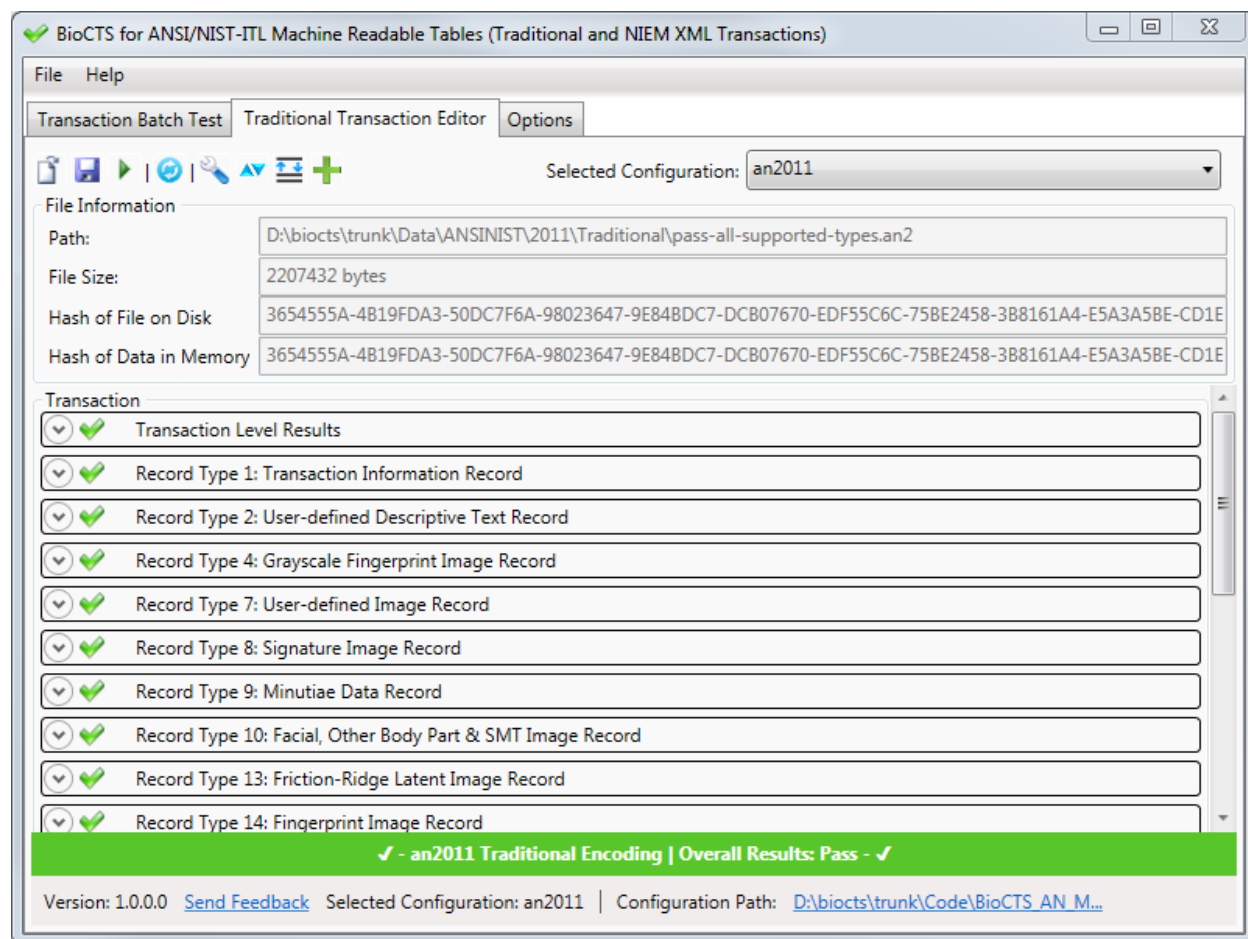

Figure 8 – Traditional Transaction Editor Overview

The Editor[6] is designed to display as little or as much data as desired by the user by making use of expander sections, which can be expanded to display more information by pressing the ⌄ button within sections. Below is a brief description of the tools[7] available in the Editor:

Open File: Loads a single file into the Editor and automatically tests it[8]

Save File: Saves the file to disk (location chosen by user) with any modifications from Editor

Test Files Tests or re-tests the file currently loaded into the Editor

Reload from Disk: Reloads the current file from disk, discarding any changes from the Editor

Auto Fix: Attempts to fix common file issues with Record Length and Transaction Contents

Sort: Sorts fields or records numerically (context dependent)

Organize/Arrange: Manually repositions fields or records using arrows (context dependent)

Add: Add field or record (context dependent)

Remove/Delete: Removes a field or record (context dependent)

Modify Subfields: Change number of subfields and information items (with or without data)

View/Modify Data: Displays data in a pop-out window and allows editing

Show Record Image: Attempts to display the image found in the record (context dependent)

The Editor displays three types of Results indicators for Records, Fields, Subfields, and Information Items:

✔ - **Passing Test Indicator** ✖ - **Failing Test Indicator** ◢ - **No Tests Results Indicator**

Expanding Transaction Hierarchy

The Editor displays the data and associated results in an expandable, hierarchical format, and allows editing of existing data using text fields. Each entity in the transaction can be expanded to reveal child entities, such as records, fields, subfields, and information items. Any data within the transaction can be edited, saved to disk, or loaded into a pop-out window for better viewing and editing. In the following image, Record Type-1 is expanded to reveal its fields, and Field 1.005 is expanded to reveal its data,

---

[6] Many Editor tools and actions, including adding or removing Records, Fields, and Subfields, cause the transaction to be re-tested automatically. In some cases, the Auto Fix feature will also be run to ensure the Transaction is formatted properly after the change takes place. For example, the removal of a Field will necessitate an adjustment to the Record Header Length to prevent parsing errors.

[7] All options for modifying or organizing fields are disabled for the first field in each record type (Field xx.001). The first field in each record is required for parsing because it specifies the record length.

[8] Drag and drop is supported on the Transaction area of the Traditional Transaction Editor tab so that files may be loaded into the editor.

which is editable. The red checkmarks indicate entities where there are failing tests. Those test results may be shown by expanding the results section.
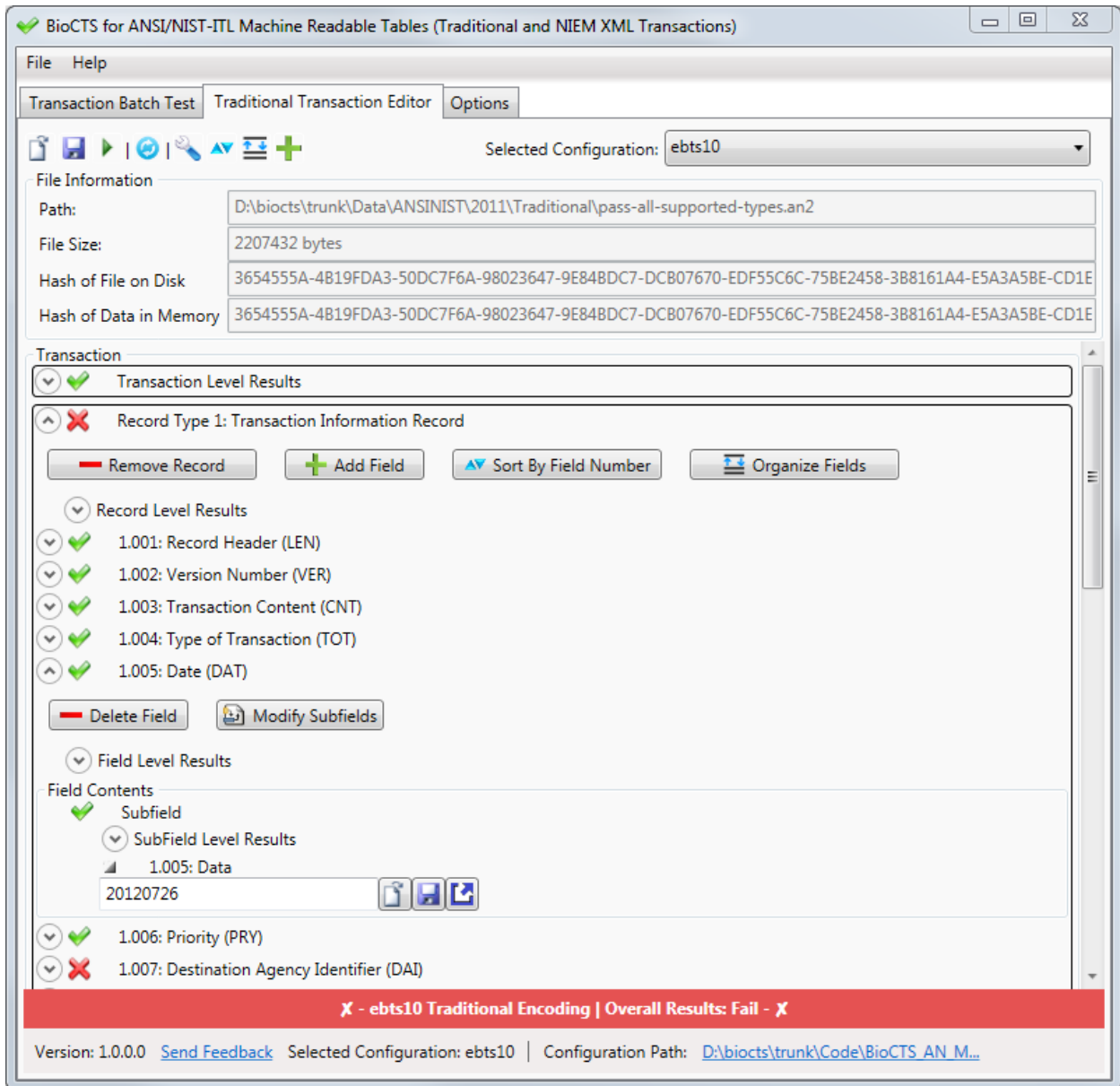
Figure 9 – Traditional Transaction Editor Expanded

Record Editing

When a record is expanded, the Editor displays a list of the fields contained in the record.

Adding and Removing

The selected or expanded Record may be deleted by clicking the "Remove Record" button. In any expanded Record, the Add, Sort, and Organize Field options are available.

New Records may be added by clicking the green plus icon ✚ . The user will be prompted for a Record Type number.
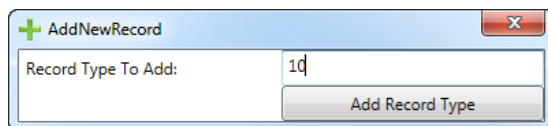
Figure 10 – The Editor Add Record Dialog

Moving and Reordering

The Records may also be reorganized within the Transaction by using the numerical sort ▲▼ or the manual rearrange ⬍ buttons.
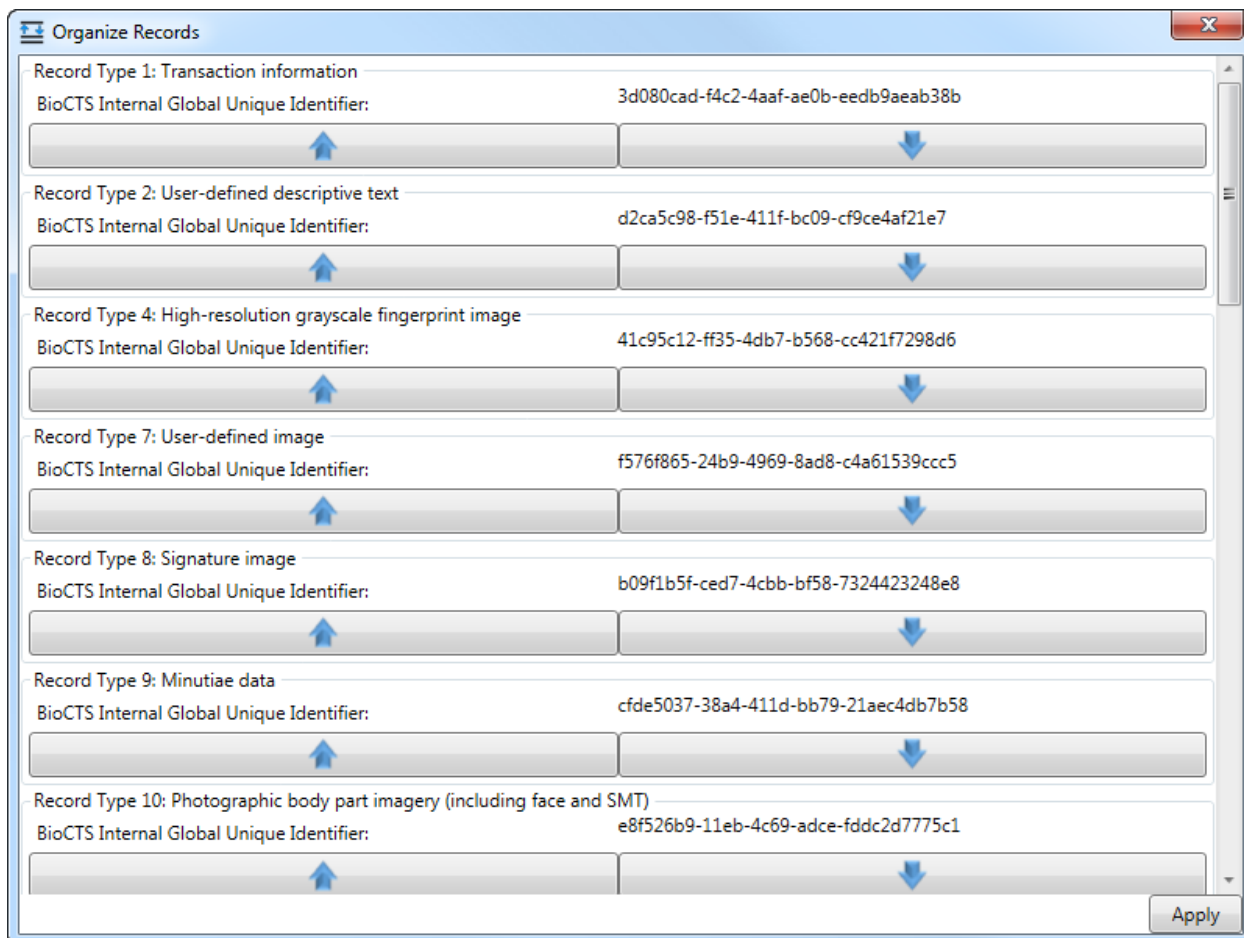
Figure 11 – The Editor Organize Records Dialog

If the selected Record contains an image, the "Show Record Image" button will be available. Clicking the button will show the image and its metadata. If the image type is not supported or cannot be read by BioCTS, a "No Preview Available" message will be displayed.
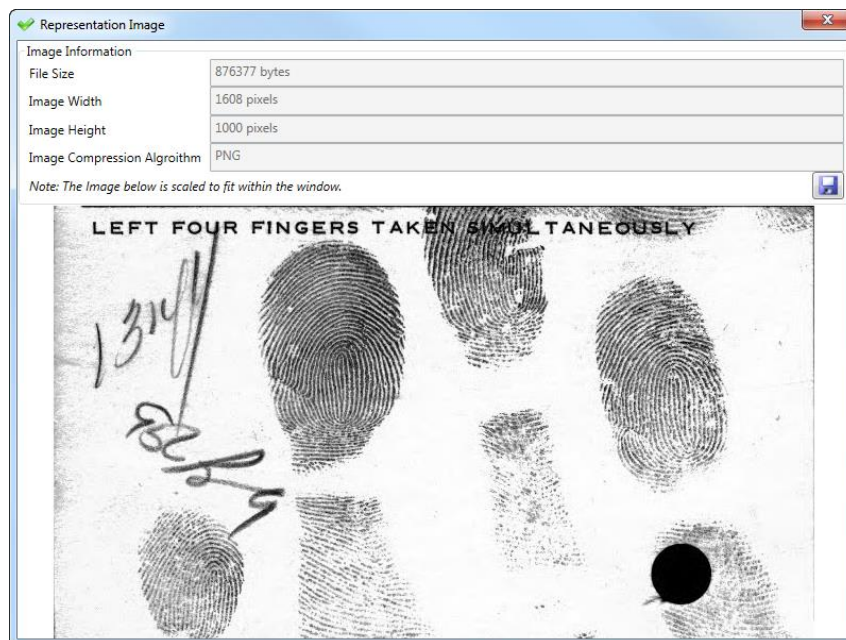
Figure 12 – Editor View Image Window

Field Editing

When expanded, a field displays expanders for the field-level results and the data that is held within the field. At this level, a Field can be deleted from a record. The values in the data text fields can be edited. Three buttons are provided to assist when a large amount of data is being manipulated (such as in an image field):

> **Load Data:** Loads data into the contents of a field/subfield/information item
> **Save Data:** Saves the contents of a field/subfield/information item to a file
> **Pop-Out Data:** Displays data in a large pop-out window for easy viewing and editing

Subfield Structures

All Fields technically contain at least one Subfield which contains at least one Information Item (see "How separators are used" in Annex B of NIST SP 500-290 Rev. 1). For Fields defined in the MRTs as having **<ContentType>** of Data, Data_T, or Data_X, the editor shows one subfield containing one information item which contains the data.

The "Modify Subfields" button allows the user to modify the number of Subfields and Information Items, and decide whether to keep the current data. In the example below, a second Information Item is being added, and the previous data is retained in the first Information Item.
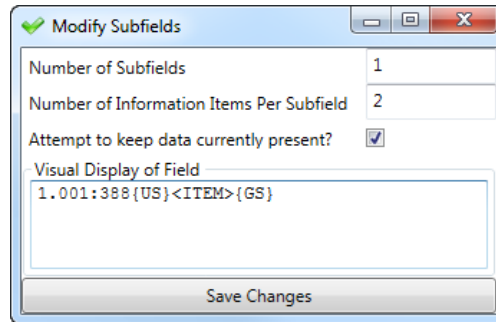
Figure 13 - The Modify Subfields Dialog

Subfield Editing

When a Field is expanded that contains multiple Subfields or Information Items, the containing Subfields and Information Items are displayed with their own set of data and Results. The Subfields and Information Item data can be edited in the same manner as Field data, by using the provided text field or the provided buttons for loading, saving, and modifying data.

Options

The "Options" tab provides the testing and output customization choices described below[9]. Note that the Options tab is disabled during testing to prevent changes that may compromise the integrity of the Test Results.
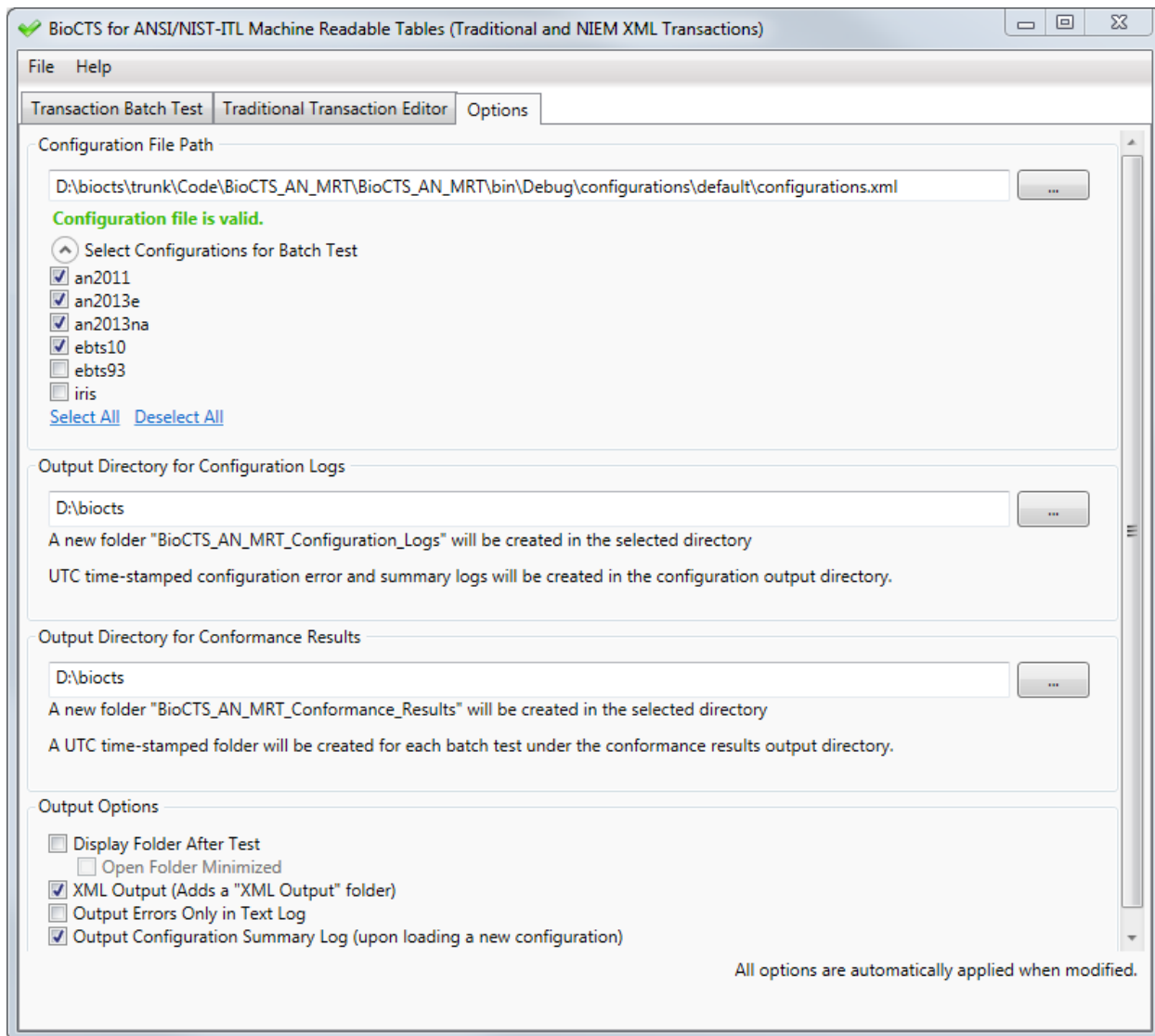
- **Configuration File Path:** This allows the user to specify the location of an MRT configuration file that provides the definitions for various configurations that describe standard and profile requirements. Upon selection, the configuration file and MRT files will be parsed and validated. A status indicator will be shown to provide information about the loading process and completion. During loading, all other user interface options are disabled. Invalid MRT files will be reported and the user will be asked to load a new file or correct the errors.

---

[9] Changes made in the Options Tab are automatically applied.

- **Select Configurations for Batch Test:** This option may be expanded to reveal a list of checkboxes indicating the configurations that may be selected or unselected for inclusion in batch testing. The columns on the Transaction Batch Test tab will be updated according to the user selections.
- **Output Directory for Configuration Logs:** The Options tab also includes a directory selection of where BioCTS saves MRT configuration log files. Within this directory, a new folder "BioCTS_AN_MRT_Configuration_Logs" will be created. UTC time-stamped error and summary logs will be created in the directory.
- **Output Directory for Conformance Results:** The Options tab also includes a directory selection of where BioCTS saves log files. Within this directory, a new folder "BioCTS_AN_MRT_Conformance_Results" will be created. A UTC time-stamped folder will be created for each batch test under the conformance results output directory. Within that time-stamped folder, the following will be created (unless indicated otherwise by the user):
  - Summary Output
  - Text Output
  - XML Output
- **Output Options:** Options exist to display the Log folder after a test and the option to generate XML logs or not. An option is available to force the Log Folder to be opened minimized to avoid interrupting the user with a pop-up window. An option is also available to output only errors in the text logs. Finally, there is an option to enable or disable output configuration summary logs when loading new MRT configurations.

BioCTS AN MRT CLI

This section describes the BioCTS AN MRT CLI Edition V.1.0.0, which provides a flexible command-line interface for testing AN-ITL transactions.

Running BioCTS MRT CLI Edition

To run BioCTS MRT CLI, use of the Command Prompt is required. It can be found by visiting the following from the Windows start menu:

Start>All Programs>Accessories>Command Prompt

The Command Prompt can also be accessed using the Windows search function to search for "Command Prompt" or "cmd".

Once a Command Prompt is running, change directories to the location where BioCTS MRT CLI is located using the "cd [directory]" command.

Run the application by typing BioCTS_AN_MRT_CLI.exe and pressing enter.

Input

Input is supplied via command line arguments, but is optional. If no input is provided, the default settings are used. The main input components are:

- AN MRT Configuration file: Specifies a set of MRT configurations, including the location of all MRT documents.
  Default: "configurations.xml"
- Configuration Name(s): The name or names of one or more configurations contained in the configurations file, indicating those to be tested.
  Default: all configurations in the configurations file.
- Source Path: The path to an ANSI/NIST-ITL transaction to be tested or a path to a directory containing ANSI/NIST-ITL transactions to be tested.
  Default: ../[appdir]/testfiles

Processing

The following generalized processing steps take place after executing the application.

- Parses the configurations file and generates a list of configurations.
- Logs configuration errors.
- Reads in the list of files (transactions) to be tested.
- Tests input files against the specified configuration(s).

Output

The following information is output after running the application:

- Summary of discovered configurations and error logs if errors are present.
- Log files containing results of the conformance testing for each input file for each configuration selected.

- A summary log of all files tested for each configuration.
- Individual summary logs for each of the files tested.

Features

BioCTS MRT Edition has many command line arguments. To view them while the program is running, type: `BioCTS_AN_MRT_CLI.exe -help`

All arguments may be omitted, indicating that BioCTS AN CLI should use the following defaults:

- Configuration file name `configurations.xml`
- Build all Configuration names held within the configuration file
- Test all AN-ITL Transactions found in the `testfiles` folder within the application folder

The program flow can be altered by passing in additional arguments. The following flags are valid for BioCTS MRT Edition:

General Flags

- `-help`
  - Displays the help file.
- `-h`
  - Displays the help file.
- `-q`
  - Quiet output. No output files or directories are opened after creation.

Conformance Flags

- `-tpath "filePath"`
  - Path to a transaction file. Flag must be followed by a file path.
- `-tdir "directoryPath"`
  - Path to a directory containing transactions. Flag must be followed by a directory path.
- `-rdir "directoryPath"`
  - Specify output directory path for the test results. Flag must be followed by a directory path.
- `-tconfig "configurationName"`
  - Test configuration to test transaction against. Only the configuration with the same name as the `configurationName` parameter will be used for testing. (Note: this flag may be repeated to specify additional configurations).
  - Values for "`configurationName`" are case-sensitive and must match the value of the config's name attribute in the configurations file. All invalid values are ignored.
- `-tprompt`
  - Prompts for Conformance Flags after loading all MRT settings and then tests for conformance. The prompts will continue until "exit" is specified. All other Conformance Flags submitted with `-tprompt` are ignored. When prompted, only Conformance Flags are valid. Previously entered MRT Flags will be respected.

- `-qr`
  - Quiet results output. Output summary directory is not opened after creation.
- `-notext`
  - The text result logs are not output to a file
- `-noxml`
  - The xml result logs are not output to a file
- `-norsum`
  - The results summary log is not output to a file

MRT Flags
- `-configpath "filePath"`
  - Path to the configuration file. Flag must be followed by a file path.
- `-logdir "directoryPath"`
  - Specify output directory path for summary and errors. Flag must be followed by a directory path.
- `-spath "filePath"`
  - Specify output file path for summary only. Flag must be followed by a valid file path.
- `-epath "filePath"`
  - Specify output file path for errors only. Flag must be followed by a valid file path.
- `-nolog`
  - No output is logged to files.
- `-noslog`
  - The summary is not logged to a file.
- `-noelog`
  - The errors are not output to a file.
- `-notime`
  - No timestamp is added to the output file names.
- `-nostime`
  - No timestamp is added to the summary file name.
- `-noetime`
  - No timestamp is added to the error log file name.
- `-qs`
  - Quiet summary output. Output summary file is not opened after creation.
- `-qe`
  - Quiet errors output. Output errors file is not opened after creation.
- `-qp`
  - Quiet progress output. Processing messages not shown in console.
- `-cout`
  - Console output. Summary and errors are shown in the console.
- `-couts`
  - Console summary. Summary is shown in the console.
- `-coute`
  - Console errors. Errors are shown in the console.

- `-eoshort`
    - Error options: shorten file paths of error locations in error reports.
- `-soc "configurationName"`
    - Summary options: configurations to output. This option is used to limit the output of the summary file.
    - By specifying this flag, only the configuration with the same name as the `configurationName` parameter will be reported in the summary file (Note: this flag may be repeated to specify additional configurations).
    - Values for "`configurationName`" are case-sensitive and must match the value of the config's name attribute in the configurations file.
- `-sod "documentTypeList"`
    - Summary options: documents to output.
    - This option requires a comma-separated list of document types to be specified, which limits the document types that will be reported for all configurations.
    - The default option is to output all document types.
    - Valid values for "`documentTypeList`" are (case insensitive):
        - `FieldDefinitions`
        - `LookupCodes`
        - `RecordTypeDesc`
        - `TOTRecords`
        - `TOTFields`
        - `TOTFieldDetails`
        - `knownErrors`
        - `fixes`
- `-socd "configurationName" "documentTypeList"`
    - Summary options: configurations to output and documents to output.
    - This option is the same as `-soc`, but also requires a comma-separated list of document types to be specified, which limits the document types that will be reported for the specified configuration.
    - Note: this restricts the output to the specified documents for the configuration, regardless of whether or not `-sod` is used. See `-sod` and `-soc` for valid values for "`configurationName`" and "`documentTypeList`".
- `-soh`
    - Summary options: Report update history in summary file (false by default).
    - This option displays an update history for the contents of configurations when they are updated by child configurations. This applies to all configurations.
- `-sohc "configurationName"`
    - Summary options: Report update history in summary file for the specified configuration.
    - Note: this option does not automatically turn on output for the specified configuration; `-soc` or `-socd` must be used to specify that the summary should report the configuration contents (unless all configurations are being reported, which is the default option).
- `-sono`

- o Summary options: no configuration output. Do not report configurations contents in the summary (false by default).
- o This option limits the summary output so that it does not report configuration contents, such as document and element details. The summary will still provide an overview of the configurations document and the relationship among configurations.

Default Settings: Conformance Results

The application reads and tests all transactions located in the default test files directory located at:
../[appdir]/testfiles

All results are written to files in the default results directory under a time-stamped folder. Text results are written to "Text Output" and XML results are written to "XML Output". For every configuration being tested, a directory of the configuration name is placed in the output directory, which contains the result logs. Finally, a "Summary Output" directory is created to hold the overall results summary (_BatchResultsSummary.txt) and a listing of results for each individual file. For example:

  ../[appdir]/[resultsdir]/[timestamp]/Text Output/[configname]/txname.txt
  ../[appdir]/[resultsdir]/[timestamp]/XML Output/[configname]/txname.txt
  ../[appdir]/[resultsdir]/[timestamp]/Summary Output/_BatchResultsSummary.txt
  ../[appdir]/[resultsdir]/[timestamp]/Summary Output/file_ResultsSummary.txt

- **[appdir]** is the directory where the application is located
- The default **[resultsdir]** name is BioCTS_AN_MRT_CLI_Conformance_Results
- **[configname]** is the name of the individual configuration
- The results directory is opened on creation
- **[timestamp]** is a UTC formatted date and time indicating the test completion datetime

Default Settings: MRT Validation

The application reads from the "configurations.xml" file in the default configurations directory at:
../[appdir]/configurations/default/configurations.xml

All output is written to files in the default output directory at:

  ../[appdir]/[logdir]/_SummaryLog_[configfilename]_[timestamp].txt
  ../[appdir]/[logdir]/_ErrorLog_[configfilename]_[timestamp].txt

- **[appdir]** is the directory where the application is located
- The default [**logdir**] name is BioCTS_AN_MRT_CLI_Configuration_Logs
- **[configfilename]** is the name of the AN MRT configuration file
- **[timestamp]** is a UTC formatted date and time indicating the test completion datetime
- All output files are opened on creation

- The summary and errors are not displayed in the console

References

NIST Special Publication 500-290 Data Format for the Interchange of Fingerprint, Facial & Other Biometric Information
https://www.nist.gov/itl/iad/image-group/ansinist-itl-standard-history

NIST Special Publication 500-290 Rev1 (2013) Data Format for the Interchange of Fingerprint, Facial & Other Biometric Information
https://www.nist.gov/itl/iad/image-group/ansinist-itl-standard-history

FBI Electronic Biometric Transmission Specification & Technical and Operation Update #5
https://www.fbibiospecs.cjis.gov/EBTS/Approved

Annex G Machine Readable Tables & Documentation
https://www.nist.gov/itl/iad/image-group/ansinist-itl-standard-references

MRT Definitions for use with BioCTS_v1.9.1
https://www.nist.gov/itl/csd/biometrics/biocts-machine-readable-tables

MRT files and related Schema for BioCTS AN MRT applications
https://www.nist.gov/itl/csd/biometrics/biocts-machine-readable-tables

BioCTS AN Sample Transaction Data
https://www.nist.gov/itl/csd/biometrics/biocts-machine-readable-tables

APPENDIX A: MRT Rules and Requirements

The document "MRT Definitions for use with BioCTS_v1.9.1" describes most of the elements of MRT documents. Many MRT features are self-explanatory, but some details are not obvious. It is important to understand the MRT definitions correctly before using them in the BioCTS software. This section outlines how the BioCTS software interprets features of the MRTs that the documentation and other resources may not clearly explain, and explains how specific constructs can be represented using the MRT elements. To look up specific MRT elements, see the MRT Elements section; to look up how to represent specific requirements, see Specifying ANSI/NIST-ITL Requirements or Specifying EBTS Requirements.

Specifying ANSI/NIST-ITL Requirements

This section explains how some specific constructs and requirements can be represented using the MRT elements.

User-defined Fields (UDF)

User-defined fields (UDFs)[10] are fields that are not defined by the standard (or the MRTs), but are still valid if present in the transaction. UDFs are specified in the MRTs by using the **<Mnemonic>** value "UDF". The associated **<FieldID>** is used to determine which field numbers or XML Paths are user-defined. For example, if **<Mnemonic>** is UDF, the following **<FieldID>** values represent the field numbers:

- **<FieldID>** value 10.200_10.900 indicates that fields 10.200 through 10.900 are user-defined.
- **<FieldID>** value 10.200 would indicate that field 10.200 is user-defined.

If a **<FieldDef>** element is defined with a **<FieldID>** that corresponds to a defined UDF field number (and the **<Mnemonic>** is not UDF), the UDF is ignored and the **<FieldDef>** is used to define the field requirements. For example, In the cases above, if a non-UDF **<FieldDef>** is defined for 10.200, that field would not be considered as user-defined; it would instead be parsed per the **<FieldDef>** specified.

Due to the nature of the binary encoding, only one user-defined field can be parsed, and only if there is data remaining after all other defined fields are parsed. The binary UDF uses the first UDF field number listed and will parse all remaining binary data to the end of the record. For example, **<FieldID>** value 07.003_07.999 indicates that any binary field in the range 7.003 to 7.999 is user-defined. If **<FieldDef>** values are present for fields 7.001 and 7.002, then any data remaining after the first two fields will be parsed as user-defined field 7.003-UDF. Note: if **<FieldDef>** 7.003 and 7.004 are defined, for example, then any remaining data beyond the first four fields would be parsed starting with the next UDF value (7.005-UDF).

---

[10]UDF fields may be defined by the receiving agency, by the domain listed in Field 1.013 Domain name / DOM, or by the application profiles listed in Field 1.016 Application profile specifications / APS. If definitions are provided in the MRTs, they override the UDF declaration. If not, the fields in the UDF range are treated as user-defined fields with no specific testing provided.

For all encodings and formats, if UDF values are not defined by the MRTs, and a field is present that is not specified by the **\<FieldDef\>** values, it is reported as an undefined field and is an error. The mnemonic for undefined Fields is represented with three questions marks (???).

For XML, the XMLPath defined by the **\<FieldID\>** with Mnemonic "UDF" indicates that the element is user-defined and should not be present in the transaction. Due to the nature of XML, all user-defined values must be substituted with a new element that is defined by the schema and the MRT.

User-defined Subfields, Information Items, and Child Elements

User-defined subfields and information items are represented in the MRTs by any **\<FieldDef\>** where the **\<InfoItem\>** value is "SET" and **\<InfoItemCount\>** is not defined or empty (this applies to XML as well). This indicates that any subfields and information items contained in the field are user-defined[10] and are not defined in the MRTs.

Due to the nature of XML, when child elements are user defined in this manner, the parent element is also considered to be user-defined. The abstract element name provided in the MRT for the parent must not be present in the transaction (they must instead be substituted with a valid element defined by the Schema).

One XML Element Representing Many Traditional Entities

Due to the nature of XML, some instances of the AN-ITL standard provide different representations of the same entities in XML when compared to the Traditional encoding. One case is when one XML element represents many Traditional entities. In this case, the **\<TechNotes\>** element is used to indicate which Traditional elements are represented by including an **AppliesTo{***ListOfEntities***}** entry, where *ListOfEntities* is a comma-separate list of traditional entities that are represented by the XML element. Possible list entries include:

- Field(*FieldID, Mnemonic*)
- InformationItem(*FieldID, Mnemonic*)

If the XML element instead represents any of a list of repeating information items in the Traditional encoding (where an information item may repeat one or more times), then the *ListofEntities* value must take the form:

RepeatingInformationItems{*ListOfInformationItems***}** where *ListOfInformationItems* is a list of comma-separated information item definitions in the form: (*FieldID, Mnemonic*).

Some examples include:

```
<TechNotes>
     AppliesTo
     {
          RepeatingInformationItems
          {
               (10.043-B,TC2),
               (10.043-C,TC3),
               (10.043-D,TC4),
```

```
                    (10.043-E,TC5),
                    (10.043-F,TC6)
            }
        }
</TechNotes>

AppliesTo
{
    InformationItem(02.091-B,RCN1),
    InformationItem(02.092-B,RCN2)
}

AppliesTo
{
    Field(02.089,MSC),
    InformationItem(02.033-G,MSC)
}
```

Specifying EBTS Requirements

This section explains how some EBTS-specific constructs and requirements can be represented using the MRT elements.

TOT Record Set Occurrence Requirements

This section details requirements related to the number of allowed record-type occurrences for each TOT value. Examples of these types of requirements can be found in Appendix L of the EBTS 10.0 specification.

In the MRTs, these requirements are specified using the TOTRecords MRT document which contains a list of **<TOTRecord>** elements in the form:

```
<TOTRecord>
    <TransactionFullName>[TOT FULL NAME]</TransactionFullName>
    <TOT>[TOT VALUE]</TOT>
    <TOT_Description>[DESCRIPTION VALUE]<TOT_Description>
    <RecordNum>[RECORD NUMBER VALUE]</RecordNum>
    <Record_Min>[RECORD MINIMUM OCCURRENCE VALUE]</Record_Min>
    <Record_Max>[RECORD MAXIMUM OCCURRENCE VALUE]</Record_Max>
</TOTRecord>
```

For the given TOT, individual record type occurrence limits are defined by referencing the **<TOTRecord>** element that specifies the related **<TOT>** and **<RecordNum>** values. The **<Record_Min>** and **<Record_Max>** values will contain the record occurrence restrictions for the specified **<RecordNum>** and **<TOT>** combination.

Duplicate **<TOTRecord>** entries are allowed, meaning that the **<RecordNum>** and **<TOT>** combination are not necessarily unique among entries. In such cases, the record type occurrence requirement is derived by compiling the information from all **<TOTRecord>** elements that have the **<TOT>** and **<RecordNum>** values of interest. The following example is for illustration only, and does

not correspond to a known record type. Each value represents the **<Record_Min>** and **<Record_Max>** values separated by a dash.

TOTRecord Entries	T1	T2	T4	T7	T9	T10	T13	T14	T15	T17
1st <TOTRecord>	1-1	1-1				0-1		0-14		1-1
2nd <TOTRecord>	1-1	1-1						1-20		0-1
TOTRecord Consolidated	1-1	1-1				0-1		0-20		0-1
Note: 1st and 2nd <TotRecord> **must have the same** <RecordNum> **and** <TOT> **values to be consolidated.**										

<p align="center">Figure 15 – Consolidated <TOTRecord> Definitions</p>

For cases where there are conflicting Record Type requirements, such as the type-14 record in the example above, the lowest minimum and the highest maximum are used to ensure the widest requirement definition.

BioCTS only tests TOT Record Set Requirements if the following conditions are true:

a) The transaction has a TOT value.
b) A TOTRecords document is defined for the related MRT configuration.
c) The TOTRecords document contains at least one TOTRecord entry with the specified TOT value.

MRT Elements

This section outlines how the BioCTS software interprets elements of the MRTs that the documentation and other resources may not clearly explain.

FieldDefinitions

<CondCode>
- Fields (XML and Binary): the **<CondCode>** specifies **field presence**. If the **<CondCode>** is Mandatory (M), then at least once occurrence must be present; otherwise none are required.
- Fields (Traditional Tagged): the **<CondCode>** specifies **field presence and field occurrence**. If the **<CondCode>** is Mandatory (M), then the occurrence is required to be 1; otherwise it is 0 to 1.
- Information Items (Traditional Tagged): the **<CondCode>** specifies **information item data presence**. If **<CondCode>** is Mandatory (M), then the data for that information item must be present for every occurrence, otherwise no data is required. [11]

<MaxOccur> and **<MinOccur>**

- Fields (XML and Binary): the **<MinOccur>** and **<MaxOccur>** elements determine the **field occurrence** limits.

---

[11] Information item separators (not data) are required to be present for all information items (see **<InfoItemCount>** for information item separator requirements).

- Fields (Traditional Tagged): the **\<MinOccur\>** and **\<MaxOccur\>** elements apply to the **subfield occurrence** and not the field occurrence (see **\<CondCode\>** for Tagged Field occurrence).
- Information Items: the **\<MinOccur\>** and **\<MaxOccur\>** elements determine the **information item occurrence** limits, where occurrence is defined by an information item that has data (otherwise it is considered to be not present).

\<InfoItemCount\>

- XML: There is no direct equivalent of information items in the XML format. Information Item information is directly related to the Traditional format, and the XML structures may vary greatly from their Traditional equivalents. Still, this number indicates the number of FieldDefs defined that are logically considered children of the current XML field, regardless of the actual location of those elements, which is defined by **\<XMLPath\>**.
- Fields (Traditional Tagged): the **\<InfoItemCount\>** indicates the number of expected **information item separators** per subfield. If the **\<InfoItemsVary\>** value is "NUM", then **\<InfoItemCount\>** indicates the minimum number of information item separators expected.
- *NOTE: **\<InfoItemCount\>** should only be used when the **\<InfoItem\>** value is "SET".*

\<InfoItem\>

- When the value is "SET", it indicates a grouping (field or subfield) that has defined children. Those children are Information Items in the Traditional encoding and XML elements in the XML encoding.
- When the value is an alphabetic entry, such as A, B, ZZ, etc. it represents a single child of the associated parent for which the **\<InfoItem\>** value is "SET". This child is an Information Item for Traditional encoding and an XML element for XML encoding. The children are listed in alphabetic order to indicate their parsing order in a transaction.
- When the value is empty or missing (not defined):
  - Fields (XML): If **\<ContentType\>** is Data or Data_X, this represents an XML element that contains data. Otherwise, if **\<ContentType\>** is Set or Set_X, this represents an XML container structure not found in the Traditional encoding.
  - Fields (Traditional Tagged): This represents a "Data Field" with 1 subfield containing 1 information item that contains data. *Note: **\<ContentType\>** should always be Data or Data_T if **\<InfoItem\>** is not defined for Traditional encoding.*

General Elements

\<RemoveRow\>

- MRT documents are designed to layer, so that only new definitions need to be defined. For example, ANSI/NIST-ITL 2013 can layer on top of ANSI/NIST-ITL 2011, defining only new requirements and fields. **\<RemoveRow\>** is used to indicate that a previously defined MRT element should be removed.
  - Note 1: MRT elements are updated regardless of whether or not **\<RemoveRow\>** is present in cases when an MRT element is redefined in a child document.

- o Note 2: BioCTS tracks the history of each MRT element as they are updated by each document. If an element is removed with **`<RemoveRow>`**, then the history of the element is not saved in its replacement. However, if a new element is specified without removing the old, then the history is saved.

Static (Non-Modifiable) Requirements

The use of MRT files as input to BioCTS provides a great amount of flexibility to the end user to specify valid fields and records in the transaction as well as their related value constraints. However, the MRT files define formats based on the ANSI/NIST-ITL line of standards, meaning that the MRTs cannot modify some foundational aspects of the standard's requirements. This section provides a list of requirements that the user cannot modify by editing the MRTs.

Parsing Requirements:

The software reports these results as "Parse" level results and will also provide a Test Reference with the text "This is a static requirement for all versions of the base standard."

Parse-Level Requirements (Non-Modifiable)			
Name	Requirement	Notes	Assertions
Data Presence	The transaction file cannot be empty (0 bytes) and must contain data in a format that represents logical record and fields.		Transaction-Data-Present Transaction-Data-Parsed
Transaction Size Limit	A single transaction can have at most 1000 Records.		Transaction-Size
Record Types	If not otherwise specified by the RecordTypeDesc document, the following defaults are used: • Tagged Record Types: 1, 2, 9 to 99. • Binary Record Types: 4, 7, 8. • Deprecated: 3, 5, 6. RecordTypeDesc can be used to define the list of valid Record Types. If any RecordDesc is listed then the defaults above no longer apply, and only the listed Record Types are considered to be valid (with the exception that, unless otherwise defined,	The base standard specifies the valid Record Types for each version.	Transaction-Records-Deprecated Valid Record Type

	3, 5, and 6 are always Deprecated). The encoding of the Record Type cannot be changed. 4, 7, and 8 are always considered to be binary (as well as 3, 5, and 6 if they are defined in the MRTs as valid types). All remaining valid Record Types are considered to be Tagged. Note: Record Type 1 is always considered to be valid. For example, if RecordTypeDesc defines records 2, 3, and 10, then the set of valid records would be: Tagged: 1, 2, and 10. Binary: 3. Deprecated: 5 and 6. Invalid: All others.		
IDC Fields	Any field with FieldNumber = 2 will be treated as an IDC value, except for those Record Type 1.	The standard specifies that the Field with FieldNumber = 2 represents the IDC in every Record Type except Type-1.	Transaction-CNT-REC-IDC-Matches-Records
Transaction Content Field	The field with FieldNumber = 3 in Record Type 1 is used to determine the parsing order and IDC values for all Records contained in the transaction. This field is required to be present.	The standard specifies that Field 1.003 defines the Transaction content.	Transaction-Content-Field-Present Transaction-CNT-REC-IDC-Matches-Records
Transaction Content Record Lengths	The size of the transaction must be equal to the sum of the record lengths specified by each Record Header (FieldNumber = 1 in each Record Type).		Transaction-Record-Lengths-Equal-TransactionSize
XML Record Category Code	The Record Category Code must be defined as having	The standard specifies the XML path for each Record Category	

	a FieldID with FieldNumber 1.	Code, and considers it to be the first field in each record.	
Traditional Record Order	Record Type 1 must occur first. The remaining records must appear in the order specified in Field 1.003. The IDC values of those records must also match the values specified in Field 1.003.	The standard specifies that Record Type 1 occurs first and contains information necessary for parsing and understanding the rest of the Transaction.	Transaction-Type1-First Transaction-CNT-REC-IDC-Matches-Records
Traditional Field Order and Presence	The first two fields are required for every Record Type. The first field in all Record Types must have FieldNumber = 1 and contain the Record Length. The second field in all Records (except Record Type 1) must have FieldNumber = 2 and contain the IDC value. When a field with FieldNumber = 999 is included in a Record, it must be the final field. Note: for binary-encoded records, the first 4 bytes is used to represent the record length (field 1) and the next byte is used to represent the IDC value.		Type-[X]-Field001IsFirst Type-[X]-Field002IsSecond
Tagged Field Format	Tagged fields must adhere to the format described in the standard, which specifies that the format for the tag shall consist of: • The logical record type number followed by a period ".", which must equal the Record type it is contained in • A field number followed by a colon ":"	The format for each field shall consist of	[FieldID]-FieldNumberMatchesRecordType [FieldID]-FieldNumberLengthOneToNine Note: The use of the colon, period, and separators is not directly tested by an assertion, but they are used to parse the transaction.

	• The field number may be any one to nine-digit number occurring between the period "." and the colon ":" The colon is followed by the information appropriate to that field; the information data makes use of the information separators described in the base standard to represent subfields and information items.		
Record Presence	There must be one and only one Record Type-1 in the Transaction. There must be at least one other Record Type in addition to the Type-1 Record.	The MRTs can be used to specify MinOccur and MaxOccur for each Record, but requirement regarding Record Type-1 and one additional type will always be tested.	Transaction-Type-1-Required Transaction-Required-Additional-Record
Record Type-1 ASCII Only	The data of Fields in Record Type-1 may only contain 7-bit ASCII values.		Transaction-Type1-ASCII
Binary Record Types	Binary Record Types are limited to Types 3 through 8.		Note: Not directly tested by an assertion, but these records are only parsed as binary.
Character Type Definitions	The Alphanumeric (A) Character Type does not include spaces. Spaces must be defined using NonPrintChar (\s)	The definition of the Alphabetic Character Type (A) was modified starting with the 2013 Update to the base standard. To avoid conflicting definitions, the 2013 version is used for all MRTs.	[FieldID]-CharType
Xml Schema Files	Xml Schema Files must be specified by the Configuration File using **<SchemaDir>**. If no path is specified, the XML		SchemaValidation

Setting	Assumption	Notes
	structure will not be validated.	

Figure 16 – Static Requirements

Image Requirements

The software parses image data using the following procedures:

Setting	Assumption	Notes
Image Fields	The image field must be the final field in the Record and: • For Non-Binary Record Types must have FieldNumber = 999 and RecordType >= 10 • For Binary Record Types the RecordType and FieldNumber are not important	The standard reserves Field 999 in Record Types 10 and higher for image data. If the data is not an image (for example DNA records), BioCTS will simply not parse an image for that Record.
Image Dimensions	The image width is determined by the field with Mnemonic = HLL and the height is determined by the field with Mnemonic = VLL.	The standard specifies that the HLL Mnemonic indicates horizontal line length and the VLL Mnemonic indicates vertical line length. These fields are represented by FieldNumbers 6 and 7 respectively, but there is no requirement indicating that this must be the case for future Record Types or those defined for profiles.

Figure 17 – Image Processing

APPENDIX B: MRT Files

This section contains information related to the MRT files released with BioCTS AN MRT. These files were based on MRTs published to the AN -ITL website in November, 2015, but modifications and other adjustments were made to account for perceived standard defects and perceived MRT defects.

Default MRTs

BioCTS MRT includes a default set of MRT files that represent many of the requirements found in the standards shown below. In order to create a configuration to test each standard, the configuration must be built by linking the related configuration to its parent as shown

Standard Name	Configuration	Parent Configuration
ANSI/NIST-ITL 1-2011	**an2011**	**(No Parent)**
ANSI/NIST-ITL 1-2013 (2013 error corrections only)	**an2013e**	**an2011**
ANSI/NIST-ITL 1-2011 Update 2013	**an2013na**	**an2013e**
FBI Electronic Biometric Specification Version 10.0.5	**ebts10**	**an2013e**
FBI Electronic Biometric Specification Version 9.3	**ebts93**	**an2013e**
FBI Electronic Biometric Specification Iris Pilot	**Iris**	**ebts10**

Figure 18 – Table of standard supported by default MRTs

For example, support for Iris is provided by building the iris configuration in following manner:

iris → ebts10 → an2013e → an2011

Note that the linking of configurations is performed automatically by BioCTS applications as long as the parents are specified in the MRT configuration file.

In order to use BioCTS to test against requirements in a specific version of the standard, these default files must be used. If user-created MRTs or modified versions of the provided MRTs are used, the test results can no longer be considered tests representing requirements of the base standards.

Standard Defects

Explanations for some of the perceived defects in the base standards are listed in this section. The default MRTs released with BioCTS MRT correct those perceived defects (see the section on MRT Modifications).

Record Header Length

The record header field in Traditional encoding, Field xx.001/LEN, indicates the length of the record in bytes. The standards specify the allowed minimum and maximum character counts for the LEN fields.

In some cases, these character count requirements are too restrictive. For example, records that are permitted to have missing images (such as Type-14) are required to have a minimum of 4 characters (1000 bytes) even though these records are often much smaller than 1000 bytes.

The following table provides an overview of the requirements for the LEN field character count among the AN-ITL standard versions, where * indicates unrestricted character length. The highlighting indicates instances of perceived defects, and the blanks indicate no specified requirement.

Rec Type	AN2011		AN2013		AN2015	
	Min	Max	Min	Max	Min	Max
1		*	2	*	1	*
2			4		1	*
4,7,8	4	4	4	4	4	4
9		8	4	8	2	*
10 and above		8	4	8	2	8

Figure 19 – Record Header Field xx.001 / LEN requirements

The minimum requirement of 4 for Record Type 2 and Record Type 9 and above is perceived as a defect because valid examples of those fields can be constructed with smaller record lengths. The smallest possible record length is shown below, and has a length of 13 characters.

$$2.1:13\boxed{GS}2.2:1\boxed{FS}$$

Figure 20 – Smallest possible record length

This indicates that the default minimum record length value, when one is not specified, should be 2 to indicate that 2 characters are needed to represent the length. Likewise, when the maximum record length value is not specified, it is considered to be unlimited. Based on these observations, the MRT tables released with BioCTS AN MRT applications include the following values (highlighting indicates different from standard):

Rec Type	AN2011		AN2013		AN2015	
	Min	Max	Min	Max	Min	Max
1	2	*	2	*	1	*
2	2	*	2	*	1	*
4,7,8	4	4	4	4	4	4
9	2	8	2	8	2	*
10 and above	2	8	2	8	2	8

Figure 21 – MRT Updated LEN Requirements

98.900 IID Character Counts

Information Item 98.900 / IID is listed in AN-ITL 1-2011 and Update 2013 as having a Min Character Count of 15 and a Max Character Count of 30. However, these values can be exceeded by valid IID values, including some of the examples provided by the standard.

The following example was included in the AN-ITL 1-2011 standard, which represents the shortest possible length of 12 characters, indicating that the specified 15 character minimum is too restrictive:

3,9.373,4,NA

The maximum of 30 characters is too restrictive given that some fields allow an unlimited number of repeating subfields, which could cause the IID value to increase indefinitely. To address this, the maximum limit of 30 should be removed.

These values were corrected to a minimum of 12 and an unlimited maximum in the 2015 update. Those values are used by the MRTs released with BioCTS MRT, beginning with AN-ITL 1-2011. Note that this solution assumes the field number will be represented using 3 or more digits. The minimum of 12 characters enforces this assumption.

MRT Modifications

During development and testing of BioCTS AN MRT, modifications were made to the MRTs and XML Shema files originally published to the ANSI/NIST-ITL website in November, 2015. These modifications account for perceived standard defects and MRT defects. All modifications are documented in the MRT files released with BioCTS AN MRT applications and labelled with the XML Comment `<!—BioCTS [Comment]-->`.