NIST Special Publication 500-304

Conformance Testing Methodology Framework for ANSI/NIST-ITL 1-2011 Update: 2013, Data Format for the Interchange of Fingerprint, Facial & Other Biometric Information

Christofer J. McGinnis
Dylan J. Yaga
Fernando L. Podio

This publication is available free of charge from:
http://dx.doi.org/10.6028/NIST.SP.500-304

National Institute of Standards and Technology
U.S. Department of Commerce

NIST Special Publication 500-304

Conformance Testing Methodology Framework for ANSI/NIST-ITL 1-2011 Update: 2013, Data Format for the Interchange of Fingerprint, Facial & Other Biometric Information

Christofer J. McGinnis
Dylan J. Yaga
Fernando L. Podio
Computer Security Division
Information Technology Laboratory

This publication is available free of charge from:
http://dx.doi.org/10.6028/NIST.SP.500-304

June 2015

U.S. Department of Commerce
Penny Pritzker, Secretary

National Institute of Standards and Technology
Willie May, Under Secretary of Commerce for Standards and Technology and Director

National Institute of Standards and Technology Special Publication 500-304
Natl. Inst. Stand. Technol. Spec. Publ. 500-304, 75 pages (June 2015)
CODEN: NSPUE2

This publication is available free of charge from:
http://dx.doi.org/10.6028/NIST.SP.500.304

Reports on Information Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) stimulates U.S. economic growth and industrial competitiveness through technical leadership and collaborative research in critical infrastructure technology, including tests, test methods, reference data, and forward-looking standards, to advance the development and productive use of information technology. To overcome barriers to usability, scalability, interoperability, and security in information systems and networks, ITL programs focus on a broad range of networking, security, and advanced information technologies, as well as the mathematical, statistical, and computational sciences. Special Publication 500-series reports on ITL's research in tests and test methods for information technology, and its collaborative activities with industry, government and academic organizations.

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

National Institute of Standards and Technology
Special Publication 500-304
Natl. Inst. Stand. Technol.
75 pages

Foreword

The existence of biometric standards alone is not enough to demonstrate that products meet the technical requirements specified in the standards. Conformance testing captures the technical description of a specification and measures whether an implementation faithfully implements the specification. Conformance testing provides developers, users, and purchasers with increased levels of confidence in product quality and increases the probability of successful interoperability. Lack of conformance to the required standard(s) can, in many cases, jeopardize the expected biometric recognition performance or prevent access to the data (as well as impact the overall operational performance) since implementers may handle non-conformant records in different ways during processing.

Although no conformance test can be comprehensive enough to test all the different combinations of mandatory requirements of a standard and all possible combinations of conditional and optional characteristics that could be included in American National Standards Institute (ANSI)/NIST-ITL 2011 Update: 2013 (AN-2013) transactions, a well-designed conformance test tool that faithfully implements a standard conformance testing methodology could raise the level of confidence on the test results. Therefore, transactions tested with such a tool (and reported to be conformant to the standard), are more likely to conform to the standard.

The Computer Security Division (CSD) of NIST/ITL supports the development of biometric conformance testing methodology standards and other conformity assessment efforts through active technical participation in the development of biometric standards and associated conformance test architectures and test suites and develops these test tools to support users who require conformance to selected biometric standards and product developers interested in conforming to biometric standards by using the same testing tools available to users. Testing laboratories can also benefit from the use of these test tools. Under the conformance test software called "BioCTS", NIST/ITL CSD develops Conformance Test Architectures (CTAs) and Conformance Test Suites (CTSs) to test implementations of national and international biometric data interchange formats. The initial version of a CTA/CTS designed to test implementations of the ANSI/NIST-ITL 1-2011 Update: 2013 was recently released. These testing tools and related documentation can be found and downloaded at: http://www.nist.gov/itl/csd/biometrics/biocta_download.cfm.

Table of Contents

List of Tables and Figures

1 Introduction

This publication defines a conformance testing methodology framework (CTMF) which includes elements of a conformance testing methodology (CTM), conformance test assertions, and conformance test procedures applicable to ANSI/NIST-ITL 2011 Update: 2013 (AN-2013). It discusses three levels of conformance testing (Level 1, Level 2, and Level 3) and provides a detailed Test Assertion Syntax for describing conformance test procedures. The Conformance Test Assertion Syntax formalizes a method for representing these conformance tests using Expressions, Value-Types, Operators, and Operands.

A table-based format for documenting AN-2013 requirements and conformance test assertions is included. The table-based format indicates the association between requirements in AN-2013 and the test assertions and test procedures required to test each assertion. It includes information on the applicability of each test assertion indicating whether it only applies to the Traditional encoding as described in Annex B of AN-2013 ("T"), to the National Information Exchange Model (NIEM)-conformant Extensible Markup Language (XML) encoding as described in Annex C of AN-2013 ("X"), or to both Traditional and NIEM (XML) encoding ("B").

AN-2013 specifies a data-interchange transaction format comprised of Record Types (collections of biometric and/or forensic modality data and related metadata). The test assertion tables included in Annex C identify requirements and assertions that are required for every transaction (regardless of its containing Record Types) according to the terms specified in AN-2013. These types of requirements are referred to as transaction-related requirements and are defined as requirements that are not related to a specific Record Type. Examples of transaction-related requirements include:

- The transaction adheres to its specified encoding (Traditional or NIEM-XML) requirements.
- The transaction includes one and only one Record Type-1.
- Record Type-1 is encoded exclusively in 7-bit ASCII (for Traditional Encoding).
- Record Type-1 is conformant to the requirements specified for its fields, subfields, and information items:
    - All mandatory fields, subfields, and information items in Record Type-1 must be present (with data), and the requirements for those entities must be met.
    - Optional and dependent fields, subfields, and information items that are present in Record Type-1 must be conformant to the requirements specified for those entities.
- the transaction includes at least one other record of a type other than Record Type-1
- the transaction does not include deprecated or reserved record types or fields.

The test assertions for Record Type-1 are also documented. The requirements in Annex C are silent regarding requirements for any specific Record Type included in the transaction other than Record Type-1. Tables of AN-2013 requirements and test assertions pertaining to specific Record Types (other than Record Type-1) are not documented in this CTMF document, but plans exist to document them in separate publications (National Institute of Standards and Technology Interagency Reports/NISTIRs).

Annex D includes test notes and test exceptions that apply to requirements and assertions documented in Annex C as well as test notes and exceptions for those requirements which may be released in separate publications.

Definitions of Type A and B testing are included in Section 4. Level 1, 2, and 3 testing are discussed in Section 5.3. The tables of requirements and test assertions in this publication address only Level 1 and 2 testing and Type-A testing. Assertions for Type-B testing are not included.

The CTMF does not establish tests of characteristics (i.e., performance, acceptance, security, robustness) of products that generate the AN-2013 transactions.

2 Conformance Test Tool Characteristics

AN-2013 conformance test tools that fully implement the CTMF for testing the AN-2013 requirements are expected to implement all the requirements of Section 5, the Conformance Testing Methodology section, including the procedures defined by Level 1 and Level 2 test assertions. Such tools are also expected to be capable of testing AN-2013 implementations against the assertions specified in Annex A for the mandatory requirements in AN-2013 and any requirements associated with Optional entities that are included in the transaction. Although many constructs (such as fields and subfields) specified in the AN-2013 standard are optional, their presence in a specific transaction indicates that the requirements specified for those constructs are mandatory (see "Implementation Required" in the table headers of the Tables of requirements and assertions format in Annex A).

3 References

NIST Special Publication 500-290 Version 2 (2013), ANSI/NIST-ITL 1-2011 Update:2013, December 2013, *Information Technology:* American National Standard for Information Systems - Data Format for the Interchange of Fingerprint, Facial & Other Biometric Information, Incorporating ANSI/NIST 1-2011 Sup:Dental & ANSI/NIST-ITL 1-2011 Sup:Voice with additional new material Available at http://biometrics.nist.gov/cs_links/standard/ansi_2012/Update-Final_Approved_Version.pdf

JPEG (Joint Photographic Experts Group), *JPEG File Interchange Format, Version 1.02*. Available at http://www.jpeg.org/public/jfif.pdf

ISO/IEC 15444-1, *JPEG 2000, Information Technology - Digital Compression and Coding of Continuous-Tone Still Images Part 1: Requirements and Guidelines*.

ISO/IEC 15444-2, *Information technology — JPEG 2000 image coding system*: *Extension*, available at: http://www.jpeg.org/metadata/15444-2.PDF

ISO/IEC 15948:2004 *Information Technology -- Computer graphics and image processing* -- Portable Network Graphics (PNG): Functional specification.

IAFIS-IC-0110 (V3.1) *WSQ Gray-scale Fingerprint Image Compression Specification*, October 4, 2010.

4 Terms and definitions

AN-2013

ANSI/NIST-ITL 2011 Update: 2013

assertion

A test procedure that represents a specific aspect of a requirement found in the base standard. The assertion is expressed using the test assertion syntax defined by the CTMF.

base standard

ANSI/NIST-ITL 1-2011 Update: 2013, Data Format for the Interchange of Fingerprint, Facial & Other Biometric Information, NIST Special Publication 500-290 Version 2 - Incorporating ANSI/NIST 1-2011 Sup:Dental & ANSI/NIST-ITL 1-2011 Sup:Voice with additional new material.

conformance

The adherence of an implementation to all specified requirements as defined in the base standard.

CTA

Conformance Testing Architecture.

CTM

Conformance Testing Methodology. A description of the procedures necessary to test an implementation for conformance to the requirements specified in the base standard.

CTMF

Conformance Testing Methodology Framework. The foundational specification of the format and procedures that must be utilized to properly document and test requirements according to the base standard, and therefore establish a Conformance Testing Methodology.

CTS

Conformance Testing Suite.

implementation

A specific AN-2013 transaction.

IUT

Implementation under test. The implementation supplied by a vendor to a laboratory for conformance testing.

test

Also known as a conformance test or assertion test, it is the execution of the testing procedure defined by an assertion or set of assertions in order to obtain a statement of conformance. The result of the test is a Boolean value that determines the implementation's conformity for the assertion. For a

given requirement, if all tests pass for the associated assertions, then the implementation is considered to be conformant for that requirement.

Type-A testing

Type-A conformance testing checks the conformance of AN-2013 transactions to the requirements in the base standard.

Type-B testing

Type-B testing checks the ability to use an AN-2013 transaction, for example in a software application.

5 Conformance testing methodology

The CTMF addresses only Level 1 and 2 testing and Type-A testing. Annex A lists test assertions for Level-1 and Level-2 requirements. While Level-3 requirements may be identified in Annex A, no related test assertions are documented (see "Hierarchy of conformance tests" for information regarding the three levels of conformance testing). Type-B testing is not addressed.

5.1 AN-2013 Requirements and Conformance Test Assertions

Tables of AN-2013 requirements and conformance test assertions are documented in this publication and included in Annex A. AN-2103 transaction-related requirements (defined as requirements that are not related to a specific Record Type) as well as AN-2013 requirements and test assertions for Record Type 1 are included. The tables provide the information necessary to facilitate the development of conformance test assertions and testing tools. Each AN-2013 requirement identified in this publication is associated with one or more specified test assertions which collectively expect to form the complete set of procedures required to test an implementation for conformance to that requirement. More details on these types of test assertions are included below:

- Transaction-related Requirements and Associated Test Assertions Table

  Includes AN-2013 requirements and associated test assertions related to all AN-2013 transactions, their data conventions, encodings, content, and other information not related to a specific Record. Examples include requirements and associated test assertions that describe (and document how to test) the structure and ordering of constructs that make up all AN-2013 transactions; nonexistence checks for deprecated Record Types 3, 5, and 6 as well as reserved Record Types 22 through 97; and requirements defined in Annex B, C, and G of AN-2013 related to AN-2013 transactions.

- Record Type-1: Transaction Information Record Requirements and Associated Test Assertions

  Includes AN-2013 requirements and test assertions related to mandatory fields, subfields, information items, and XML Elements in Record Type-1 that must be met for every AN-2013 transaction; optional constructs for Record Type-1 (if any optional construct is present in a transaction, the defined requirements for those constructs become mandatory); and those related

to testing that one and only one instance of Record Type-1 is present in every AN-2013 transaction.

5.2 Limitations and exceptions

Section 1 describes the AN-2013 requirements that are documented with the associated required test assertions. Complementary publications are planned to be released which will document AN-2013 requirements and test assertions for specific AN-2013 Record Types in a format that complies with the CTMF described in this publication. A comprehensive AN-2013 CTMF (for testing all Record Types specified in the AN-2013 standard) would consist of the methodology documented in this publication as well as the set of requirements and assertions for each Record Type in AN-2013.

While conformance of an implementation to all relevant requirements can be determined, no test tool is guaranteed to be comprehensive and prove that a given system generating or using AN-2013 transactions is conformant under all possible circumstances. Well-designed conformance tests can, however, test the most likely sources of problems and demonstrate non-conformity (i.e., if errors are found, non-conformance of the transaction is likely), but the absence of detected errors does not necessarily imply full conformance to the standard.

5.3 Hierarchy of conformance tests

Three levels of conformance testing are briefly described. For each assertion included in the tables of requirements and assertions, a level of conformance testing is indicated.

Level 1 conformance testing

Level 1 conformance testing deals with the form and structure of the internal content and verifies that data structures exist and have allowable values. Specifically, it checks for the presence, structure, and value of each field, subfield, and information item in a transaction for conformance with the specification of the standard, both in terms of ranges and cardinality. Since Level 1 testing can be performed by a simple field-by-field reading of the standard and comparison to known values and their encoding, only the AN-2013 transactions are required for conformance testing, and no hardware or software components are used to create those transactions.

Level 2 conformance testing

Level 2 conformance testing deals with explicit requirements that check for internal consistency. Specifically, morphological conformance checks the relationships between fields, subfields, or information items within a transaction, including comparisons of values, as specified in the AN-2013 standard. Level 2 tests involve interactions between multiple values from different parts of the standard and sometimes from implicit observations that are not explicitly stated in the base standard. Thus, Level 2 tests require more complex validation than Level 1. Similar to Level 1 testing, Level 2 conformance testing only requires an AN-2013 transaction(s).

Level 3 conformance testing

Level 3 conformance testing checks if the biometric transaction is a faithful representation of the parent biometric data and ensures requirements are satisfied that are not merely Level 1 and Level 2 tests. Individual fields may have explicit semantic requirements for which conformance testing is significantly difficult or even impossible to test. Unlike Level 1 and Level 2 testing, Level 3 testing may require software and hardware components used to create the AN-2013 transactions, and may also require the subject and samples from which the biometric information stored in the transaction was collected. The requirements and assertion tables indicate whether Level 1 or Level 2 conformance testing is required to address the assertion identified in the test assertion. Required Level 3 conformance tests are not performed but they may be identified in the tables to indicate that the requirement is not addressed or that it is not currently testable.

XML Schemas and Conformance Testing

This CTMF, where possible, leverages the conformance-related information contained in the XML Schemas specified within the AN-2013 standard; however, the XML Schemas are only part of the overall testing. Section C.5.1 in the ANSI/NIST-ITL 1-2011 Update: 2013 standard specifies:

> *To the extent possible, the schema defines data types and constraints that enforce the allowable content rules of the base standard. Nevertheless, the XML schema may not strictly enforce the allowable content. The base standard defines allowable content, and its requirements shall be met by implementers regardless of encoding method.*

Careful analysis of the XML Schemas, distributed on the ANSI/NIST-ITL Homepage Website, reveal discrepancies between the XML Schema requirements and the AN-2013 standard requirements:

- Level 1 conformance testing – Not all allowable values have been specified within the XML Schema files.
- Level 2 conformance testing – Many interrelationship, internal consistency, and interaction tests between multiple values from different sections of a transaction are incapable of being specified within the XML Schemas.

The XML Schema files may not strictly enforce the allowable content in two ways:

- By Being Overly Broad – Which allows for the testing of more values than the allowable values as specified in the AN-2013 standard requirements. If this is the case, there are additional Assertions specified in this CTMF to test for the actual range of values.
- By Preventing Requirements – This case is when the XML Schema explicitly prevents base requirements specified in the AN-2013 standard from being tested. If this is the case the only option is to modify the XML Schema files. This modified Schema file is not included in this CTMF, but will be documented separately and made available at: http://www.nist.gov/itl/csd/biometrics/biocta_download.cfm.

5.4 Functional Documentation of Requirements and Test Assertions

This section defines the syntax and format required to explicitly identify and document AN-2013 requirements and conformance test assertions in a concise manner that conveys the necessary information for conformance testing.

Test Assertion Syntax

Test Assertions represent the set of tests performed to determine conformance for a specific Requirement specified in AN-2013. The Test Assertion Syntax described in this section formalizes a method for representing these conformance tests using Expressions, Value-Types, Operators, and Operands to describe Test Assertions.

In some instances, a Test Assertion cannot be clearly or easily represented using this syntax. These cases are referred to as Complex Assertions and English is used to express the assertion in the following format: Complex (Description), where Description is a summary of the Test Assertion. Additional syntax is described for use in complex Test Assertions only to help clarify their meaning.

Test Assertions

Test Assertions are evaluated to obtain a Test Result, which may be Pass, Fail, or Warning. Pass indicates the likelihood that the implementation conforms to that specific requirement, while Fail indicates again the likelihood that the implementation does not meet that specific requirement in the standard. Warning indicates that no errors were detected but provides additional information useful for the implementer.

Test Assertions are made up of one or more Expressions as defined in the Expression Definitions table. The outermost Expression in any Test Assertion must return a Boolean or Test Result value as defined in the Value-Type Definitions table. If the outermost Expression returns a Boolean, it is converted to a Test Result in the following manner: True becomes Pass, and False becomes Fail.

Value Types

The following table lists the Value Types that may be used in the Expressions that make up the Assertion Syntax. The actual value, represented by VALUE in the table, is contained in parentheses just after the Value Type identifier. Value Types differ from Expressions, because Value Types cannot accept any operands; they accept only the defined values specified in the Valid Values column. For example, Int(-1) is the Integer value negative one.

Due to the nature of AN-2013 transactions which may include multiple instances of each record type other than Type-1, any specified entity Value Type is not necessarily unique. For example, Fld(10.001) represents every occurrence of Field 1 in any Record Type-10.

Table 5.1 - Assertion Syntax: Value-Type Definitions

Value-Type Definitions		
Value Type	**Valid Values**	**Syntax**
Boolean	True, False.	Bool(VALUE)
TestResult	Pass, Fail, Warning. Any of the values may be followed by a description in the form: Pass(description), Fail(description), Warning(description). The description is a message that should be displayed with the result. A description is optional.	Result(VALUE)
Numeric	Any rational numeric value, for example: …-2,-1,0,1,2…, 1.1, -10.01, 0x32... Note that NumericInteger and NumericByte can be used in place of Numeric in any instance, since they are of type Numeric. However, the reverse is not true. Numeric values do not have leading zeros. For example, Num(02) is invalid, and should instead be Num(2).	Num(VALUE)

NumericInteger	Any Integer Value: …-2,-1,0,1,2…,	Int(VALUE)
NumericByte	One byte of binary data, represented in Hex. The value is represented using “0x”. For example: 0x30	Byte(VALUE)
String	Any sequence of valid Unicode characters represented by text. If the string contains characters that cannot be represented visually, the Unicode expression may be used to return a string given a set of Unicode codepoint values. If a string should be verbatim (for example there is a need to represent U+0030 as a string and not the Unicode equivalent) quotes should surround the value. To represent quotes in a string, “” is used. Examples: Str(123) is equal to Unicode(Set-Str([U+0031, U+0032, U+0033])). Both represent the string “123” Str(“”U+0030””) is “U+0030” Str(“U+0300”) is U+0300 Str(U+0030) is 0	Str(VALUE)
StringList	A list of valid String Value-Types separated by the | character in the form: firstValue|secondValue|... Examples: StrList(This is|a|List) StrList(Unicode(U+001C)|1|2|Unicode(Set-Str([U+001E,U+001F])))	StrList(VALUE)
Set	Any set of values included in the [] characters. The comma is used to separate values. The “to” term is used to represent a range for Integer Numeric types. For example, Set-Num([0,2, 5 to 100]), Set-Str([A, B, C, Alfa, Bravo, Charlie]), Set-Fld([1.001, 1.002 to 1.005]), Set-Byte([0x00, 0xFF]) Note that for sets of Strings, the comma is reserved as a separator between values. To represent a comma in a Set-Str, use U+002C.	Set-TYPE(VALUE), where TYPE is any of the Value-Type Syntax representations.
EntityTransaction	The transaction that contains all other entities in the file.	Transaction
EntityRecord	A Record represented by the numeric value of the Record Type. Ex: Rec(10) is Record Type-10.	Rec(VALUE)
EntityField	The Field represented by the string in the form: RT.FN RT is Record Type, FN is Field Number Ex: 10.998 is Field 998 in Record Type-10	Fld(VALUE)
EntitySubField	The Subfield represented by the string operand in the form: RT.FN.SI RT is Record Type, FN is Field Number, SI is the Subfield Index. If no SI is specified, then this refers to any Subfield in the Field. 1.003.1 is Subfield 1 in Field 1.003, 1.003 is each Subfield in Field 1.003; this is useful when subfields are unbounded, so that an assertion may address each subfield without knowing the number of subfields that will be present.	SubF(VALUE)
EntityInfoItem	The Information Item represented by the string in the form: RT.FN.IM RT is Record Type, FN is Field Number, IM is InfoItem Mnemonic Ex: 1.003.IDC is Info Item IDC in Field 1.003	InfoI(VALUE)
EntityElement	XML Element with name indicated by the string in the form: RT.FN.XN RT is Record Type, FN is Field Number, XN is the Xml element name Ex:1.002.biom:TransactionMajorVersionValue	XElm(VALUE)
EntityContainerElement	XML Container Element (has no Field Number, and is only used to organize other Xml Elements) indicated by the string in the form: RT.XN RT is Record Type, FN is Field Number, XN is the Xml element name Ex: XElmC(10.biom:FaceImage)	XCont(VALUE)

In order to decrease redundancy, predefined values are listed here and used multiple times throughout the tables of requirements and assertions.

Table 5.2 - Assertion Syntax: Defined Values

Defined Values		
Defined Value	**Containing Values**	**Notation**
Integer Set	Set of all Integers (positive and negative integral values, including zero) represented as a Set-Int.	Integers
Numeric Leading Zero	The string value is a Regular Expression in the form: Str(^(\+|-){0,1}0([0-9]+|[0-9]+\.[0-9]+)$) The expression represents any numeric value (with or without a + or – sign) that has a leading zero followed by one or more numeric digits (which optionally may be followed by a decimal point and one or more numeric digits after the decimal).	LeadingZeroNum
Time Index	The string value is a Regular Expression in the form: Str(^([0,1][0-9]|2[0-3]):[0-5][0-9]:[0-5][0-9].[0-9]{3}$) The expression represents time in the form hh:mm:ss.nnn, where hh is the two-digit hour,mm is the two-digit minute, ss is the two-digit seconds, and nnn is the three-digit milliseconds. (See section 7.7.2.5 of the standard).	ValidTimeIdx
Date Range Estimate	The string expression is a Regular Expression in the form: ^((Y[0-9]{1,2}){0,1}(M[0-9]{1,2}){0,1}(D[0-9]{1,2}){0,1})$ The expression represents the amount of time used as an offset (plus or minus), in the form **Y**yy**M**mm**D**dd . Any of **Y**yy, **M**mm, or **D**dd may be omitted. Bold letters are constants; yy is the 2-digit year offset, mm is the 2-digit month offset, and dd is the 2-digit day offset. The bold letters are constants.	DateRangeEstimate
Date Time Range Estimate	The string expression is a Regular Expression in the form: ^((Y[0-9]{1,2}){0,1}(M[0-9]{1,2}){0,1}(D[0-9]{1,2}){0,1}(h[0-9]{1,2}){0,1}(m[0-9]{1,2}){0,1})$ The expression represents the amount of time used as an offset (plus or minus), in the form **Y**yy**M**mm**D**dd**h**hh**m**mm . Any of **Y**yy, **M**mm, **D**dd, **h**hh, or **m**mm may be omitted. The bold letters are constants.	DateTimeRangeEstimate
Numeric Characters	Set-Str([0,1,2,3,4,5,6,7,8,9])	CharNum
Alphabetic Characters	Set-Str ([A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,a,b,c,d,e,f,g,h,I,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z])	CharAlpha
Alphanumeric Characters	Set-Str ([0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z, a,b,c,d,e,f,g,h,I,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z])	CharAlphaNum
Hexadecimal Characters	Set-Str([0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F])	CharHex
Unicode Characters	Set of all Unicode characters excluding the Special Reserved Characters (CharReserved) defined in this table. (Represented as a Set-Str of single-character strings).	CharU
Base64 Characters	Set-Str([0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z, a,b,c,d,e,f,g,h,I,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,/,+,=])	CharBase64
7-bit ASCII Characters	All characters starting with Unicode(U+0000) and ending with Unicode(U+007F) excluding the Special Reserved Characters defined in this table. (Represented as a Set-Str of single-character strings).	CharAscii
ASCII-Printable	All characters starting with Unicode(U+0020) and ending with Unicode(U+007E).	CharAsciiPr

Characters		intable
Special Reserved Characters	Set-Str([Unicode(U+0002), Unicode(U+0003), Unicode(U+001C), Unicode(U+001D), Unicode(U+001E), Unicode(U+001F)]) (Special characters “STX”, “ETX”, “FS”, “GS”, “RS”, and “US”)	CharReserved
Control Characters	Set-Str([Unicode(U+0000), Unicode(U+0001), Unicode(U+0004), Unicode(U+0005), Unicode(U+0006), Unicode(U+0007), Unicode(U+0008), Unicode(U+0009) , Unicode(U+000A), Unicode(U+000B), Unicode(U+000C), Unicode(U+000D), Unicode(U+000E), Unicode(U+000F), Unicode(U+0010), Unicode(U+0011), Unicode(U+0012), Unicode(U+0013), Unicode(U+0014), Unicode(U+0015), Unicode(U+0016), Unicode(U+0017), Unicode(U+0018), Unicode(U+0019), Unicode(U+001A), Unicode(U+001B), Unicode(U+007F)]) (Control characters “NUL”, “SOH”, “EOT”, “ENQ”, “ACK”, “BEL”, “BS”, “HT”, “LF”, “VT”, “FF”, “CR”, “SO”, “SI”, “DLE”, “DC1”, “DC2”, “DC3”, “DC4”, “NAK”, “SYN”, “ETB”, “CAN”, “EM”, “SUB”, “ESC” and “DEL”)	CharCtrl
Resolution Migration Path (ppi)	Set-Int([500, 1000, 2000, 4000, 8000, 16000, 32000, 64000]) (This represents 500 ppi minimum with 100% increase and a 5-digit maximum set by the field width for THPS/TVPS)	Resolution Path-ppi
Resolution Migration Path (ppcm)	Set-Int([197, 394, 787, 1575, 3150, 6299, 12598, 25197, 50394]) (This represents 197 ppcm minimum with 100 % increase and a 5-digit maximum set by the field width for THPS/TVPS)	Resolution Path-ppcm
ISO-3166-1 Codes	Set-Str representing all values in the ISO Country Code Standard ISO-3166-1.	ISO-3166-1
ISO-3166-1 Alpha 2 Codes	Set-Str of all 2-character alphabetic codes in the ISO Country Code Standard ISO-3166-1.	ISO-3166-1-Alpha2
ISO-3166-1 Alpha 3 Codes	Set-Str of all 3-character alphabetic codes in the ISO Country Code Standard ISO-3166-1.	ISO-3166-1-Alpha3
ISO-3166-1 Numeric Codes	Set-Str of all numeric codes in the ISO Country Code Standard ISO-3166-1.	ISO-3166-1-Numeric
GENC Codes	Set-Str representing all values in the GENC Country Code Standard, Edition 2.0 (NGA.STND.0033_2.0).	GENC
GENC Alpha 2 Codes	Set-Str of all 2-character alphabetic codes in the GENC Country Code Standard, Edition 2.0 (NGA.STND.0033_2.0).	GENC-Alpha2
GENC Alpha 3 Codes	Set-Str of all 3-character alphabetic codes in the GENC Country Code Standard, Edition 2.0 (NGA.STND.0033_2.0).	GENC-Alpha3
GENC Numeric Codes	Set-Str of all numeric codes in the GENC Country Code Standard, Edition 2.0 (NGA.STND.0033_2.0).	GENC-Numeric

The following String values represent image metadata tags that are defined for use in the syntax. Their positions within an image are based upon the compression algorithm of the image, indicated by the Image Type(s) column. Table 5.3 indicates tags that contain data, while Table 5.4 indicates tags that are used as positioning or identification markers in the image. For both tables, the Image Types may also be represented by the associated numeric code from Table 15 in AN-2013. Note: for PNG values in this table, values are converted from ppm to ppi or ppcm as indicated. This syntax follows the recommendation for rounding indicated in 7.7.8.5 of AN-2013 (round up).

Table 5.3 - Assertion Syntax: Value-based Image Metadata Tags

Image Metadata		
Term	**Image Type(s)**	**Implementation**
Width	JPEG	4th parameter of the Frame Header not counting the SOF marker
	JP2	2nd parameter of Image Header box
	PNG	1st parameter of IHDR chunk
	WSQ	5th parameter of SOF not counting the SOF marker
Height	JPEG	3rd parameter of the Frame Header not counting the SOF marker
	JP2	1st parameter of Image Header box
	PNG	2nd parameter of IHDR chunk
	WSQ	4th parameter of SOF not counting the SOF marker
SamplingUnits	JPEG	4th parameter in JFIF Header not counting the APP0 Marker
	JP2	Undefined
	PNG	3rd parameter of PHYS chunk (optional)
	WSQ	Undefined
HorzDensity-ppi	JPEG	5th parameter in JFIF Header not counting the APP0 Marker
	JP2	Undefined
	PNG	0.0254 (meters/inch) x 1st parameter in PHYS Chunk (optional)
	WSQ	Undefined
HorzDensity-ppcm	JPEG	5th parameter in JFIF Header not counting the APP0 Marker
	JP2	Undefined
	PNG	0.01 (meters/cm) x 1st parameter in PHYS Chunk (optional)
	WSQ	Undefined
VertDensity-ppi	JPEG	6th parameter in JFIF Header not counting the APP0 Marker
	JP2	Undefined
	PNG	0.0254 (meters/inch) x 2nd parameter in PHYS Chunk (optional)
	WSQ	Undefined
VertDensity-ppcm	JPEG	6th parameter in JFIF Header not counting the APP0 Marker
	JP2	Undefined
	PNG	0.01 (meters/cm) x 2nd parameter in PHYS Chunk (optional)
	WSQ	Undefined
AspectRatio	JPEG	5th parameter in JFIF Header / 6th parameter in JFIF Header (not counting the APP0 Marker)
	JP2	Undefined
	PNG	1st parameter in PHYS Chunk / 2nd parameter in PHYS Chunk (optional)
	WSQ	Undefined
BPX	JPEG	2nd parameter of the Frame Header not counting the SOF marker
	JP2	7 LSB of 4th parameter of ImgBox + 1 if 4th parameter of ImgBox is not 255
	PNG	3rd parameter of IHDR chunk
	WSQ	Undefined
CSP	JPEG	Undefined
	JP2	4th parameter of Colour Specification box
	PNG	4th parameter of IHDR chunk
	WSQ	Undefined

Table 5.4 - Assertion Syntax: Marker-based Image Metadata Tags

Image Metadata		
Image Type	**Valid Image Markers**	**Implementation**
JPEG	SOI	Start of JPEG type image.
	SOF	Start of frame in a JPEG type image.
	EOI	End of a JPEG image.
	JFIF Header	Frame for specifying JPEG type image metadata. Its inclusion is required by the standard.
JP2	SigBox	Signature Box that marks the start of a JP2 type image.
	HeadBox	Header Box in a JP2 type image.
	ImgBox	Image Header Box in a JP2 type image.
	EOI	End of JP2 image.
PNG	PNGSig	Signature of a PNG image.
	IHDR	Image Header Chunk in a PNG image.
	IDAT	Image Data Chunk in a PNG image.
	IEND	Image End Chunk in a PNG image.
WSQ	SOI	Start of WSQ type image.
	SOF	Start of frame in a WSQ type image.
	EOI	End of a WSQ image.

Expressions

Test Assertions are composed of one or more Expressions, which are statements that return a Value Type as defined in the Value-Type Definitions table. Each Expression has a single Operator, a set of valid Operands, and Return Type. Expressions may serve as Operands of other Expressions as long as their Return Type is a valid Operand for that Expression.

The tables below include a complete description of the Expressions and their required Operators and Operands used throughout the requirements and assertion tables. The Operators are categorized according to the type of Expression in which they are used and the Return Type they produce. Return Types must be a valid Value Type. The tables indicate the following information for each Expression:

- *Expression Name*: the name of the Expression
- *Description*: an explanation of how the Expression is used to return the specified Value Type
- *Return Type*: the Return Type that is returned from the Expression. The Return Type must be a valid Value Type.
- *Operands Types*: the number and type of operands accepted by the Expression. In addition to the Value Type indicated, any Expression that returns that type can be used as an Operand. Underlined Operands are optional. If operands are not formatted properly, the test assertion that contains the expression has a failing TestResult.
- *Operator Syntax*: the Operator used to represent the Expression in the assertion tables

Boolean Expressions-
These Expressions return Boolean Values. There are several types of Boolean Expressions including Relational (comparison between values), Logical (Boolean logic statements), and Conditional (if/then) Boolean expressions.

Result Expressions-

These Expressions return a Test Result, meaning that the value may be Pass, Fail, or Warning. This is the only expression type capable of generating a Warning.

Numeric Expressions-

These Expressions return a Numeric value or any of its sub-types: NumericInteger or NumericByte.

String Expressions-

These Expressions return a String value.

Generic Expressions-

These Expressions return values that depend upon the Operand Type used with the expression. Each Generic Expression is intended to be used with various Value-Types.

Table 5.5 - Assertion Syntax: Expression Definitions

Expression Definitions					
	Expression Name	**Description**	**Return Type**	**Operand Types**	**Operator Syntax**
Boolean	**Equal To**	Relational Test for equality between two operands Op1 and Op2. For comparisons between String and Numeric types, the String value is converted to Numeric first (if it cannot be converted, the result is FALSE). Ex. EQ(Str(4.0), Num(4)) is TRUE Ex. EQ(Str(4.0), Str(4)) is FALSE	Boolean	Op1: Numeric or String Op2: Numeric or String	EQ(Op1, Op2)
	Not Equal To	Relational Test for non-equality between two operands Op1 and Op2. For comparisons between String and Numeric types, the String value is converted to Numeric first (if it cannot be converted, the result is TRUE).	Boolean	Op1: Numeric or String Op2: Numeric or String	NEQ(Op1, Op2)
	Greater Than	Relational Test for if Op1 is greater than Op2.	Boolean	Op1: Numeric Op2: Numeric	GT(Op1, Op2)
	Greater Than or Equal To	Relational Test for if Op1 is greater than or equal to Op2.	Boolean	Op1: Numeric Op2: Numeric	GTE(Op1, Op2)
	Less Than	Relational Test for if Op1 is less than Op2.	Boolean	Op1: Numeric Op2: Numeric	LT(Op1, Op2)
	Less Than or Equal To	Relational Test for if Op1 is less than or equal to Op2.	Boolean	Op1: Numeric Op2: Numeric	LTE(Op1, Op2)
	Range (Inclusive)	Relational Test for if Op1 is in the range of values specified Op2 and Op3, where Op2 is the minimum numeric value and Op3 is the maximum numeric value. Ex: InRange(Num(10.1), Num(10.0), Num(10.3)) returns TRUE.	Boolean	Op1: Numeric Op2: Numeric Op3: Numeric	InRange(Op1, Op2, Op3)

Expression Definitions				
Expression Name	**Description**	**Return Type**	**Operand Types**	**Operator Syntax**
Member Of	Relational Test for if the value Op1 is a contained within the set Op2.	Boolean	Op1: Numeric or String Op2: Set-Numeric or Set or Set-String Op2 Set Type must match Op1 Value Type.	MO(Op1, Op2)
Any Member Of	Relational Test for if any values in the Set Op1 are a member of the Set Op2.	Boolean	Op1: Set (any type) Op2: Set (any type) The Set type of Op1 and Op2 must be the same.	AnyMO(Op1, Op2)
Is Subset Of	Relational Test for if the Set Op1 is a subset of the Set Op2, represented mathematically as Op1 ⊆ Op2. This means that every value in Set Op1 must exist in Set Op2. If a value repeats in set Op1, only one instance is required in Set Op2. Note that this does not have to be a proper subset. One use of this Expression is to test for character types, for example: SubSet(Chars({Fld(1.002)}), CharNum)	Boolean	Op1: Set (any type) Op2: Set (any type) The Set type of Op1 and Op2 must be the same.	SubSet(Op1, Op2)
Logical And	Logical Test returns the result of the logical AND of two Boolean operands, Op1 and Op2. Returns TRUE only if both Op1 and Op2 are TRUE.	Boolean	Op1: Boolean Op2: Boolean	AND(Op1, Op2)
Logical Or	Logical Test returns the result of the logical OR of two Boolean operands, Op1 and Op2. Returns TRUE if either Op1 or Op2 is TRUE.	Boolean	Op1: Boolean Op2: Boolean	OR(Op1, Op2)
Logical Exclusive Or	Logical Test returns the result of the logical XOR of two Boolean operands, Op1 and Op2. Returns TRUE if only one of the operands is TRUE and the other operand is FALSE.	Boolean	Op1: Boolean Op2: Boolean	XOR(Op1, Op2)
Logical Negate	Logical Test returns a value that is the logical opposite of the operand, Op1. Returns TRUE only if Op1 is FALSE.	Boolean	Op1: Boolean	NOT(Op1)
Conditional If/Then	Conditional Test evaluates the conditional statement of IF Op1, THEN Op2, where Op1 and Op2 are of Boolean Value-Type. The expression returns Op2 if Op1 is TRUE and TRUE otherwise.	Boolean	Op1: Boolean Op2: Boolean	IfThen(Op1, Op2)
Conditional If/Then/Else	Conditinal Test evaluates the conditional statement of IF Op1, THEN Op2, ELSE Op3, where Op1, Op2, and Op3 are of Boolean Value-Type. The	Boolean	Op1: Boolean Op2: Boolean Op3: Boolean	IfThenElse(Op1, Op2, Op3)

Expression Definitions				
Expression Name	**Description**	**Return Type**	**Operand Types**	**Operator Syntax**
	expression returns Op2 if Op1 is TRUE and Op3 otherwise.			
Conditional If and Only If	Conditional Test evaluates the conditional statement of Op1 IF AND ONLY IF Op2. The expression is equivalent to IF Op1, THEN Op2 AND IF Op2, THEN Op1.	Boolean	Op1: Boolean Op2: Boolean	Iff(Op1, Op2)
Entity Present	General Boolean Test returns a result indicating whether the entity or set of entities represented by the operand Op1 is present. AnyPresent indicates if any of the entities in the set Op2 are present. AllPresent indicates if all of the entities are present. Note: presence of Information Items is indicated by both the information separator tag and data being present. Ex: Present(Fld(1.001)) checks if Field 1.001 is present. Present(Set-Rec([10 to 15])) checks if Record Types 10 to 15 are present.	Boolean	Op1: Entity or Set-Entity Op2: Set-Entity	Present(Op1) Or AnyPresent(Op2) Or AllPresent(Op2)
Information Item Structure	General Boolean Test that returns a result indicating whether or not the Information Items indicated by the 1-based indexes in Op2 are present (with data) in the Field or Subfield represented by Op1. If Op2 is not specified, then all Information Items must have data. Note that Op1 is permitted to be more than one subfield, for example: InfoItemsHaveData(SubFld(1.003), Set-Int([1,2])) indicates that the first and second information item in every subfield of Field 1.003 must have data.	Boolean	Op1: EntityField or EntitySubField Op2: Set-Integer	
Image Tags Match Compression Algorithm	A General Boolean Test that returns a result indicating whether the Image in the field Op2 contains the valid metadata tags (indicated in Table 5.4) for the compression type specified in the field Op1. For uncompressed images, this test always returns TRUE.	Boolean	Op1: EntityField or EntityElement Op2: EntityField or EntityElement	
Image Tags Valid	A General Boolean Test that returns a result indicating whether the Image in field Op1 contains valid image metadata tags for any of the Image Types specified in Table 5.4. For uncompressed images, this test always	Boolean	Op1: EntityField or EntityElement	

Expression Definitions					
	Expression Name	**Description**	**Return Type**	**Operand Types**	**Operator Syntax**
		returns TRUE.			
	Image Tag Value Compare	A General Boolean Test that returns a result indicating whether the Image Tag Op2 in the Image in field Op3 matches the value in field Op1. For uncompressed images, this test always returns TRUE (with a message indicating that these tests are not performed on uncompressed images). Op2 is a Term value from Table 5.3.	Boolean	Op1: EntityField or EntityElement Op2: String Op3: EntityField or EntityElement	
	Image Tag Value Compare Aspect Ratio	A General Boolean Test that returns a result indicating whether the Aspect Ratio (Table 5.3) in the Image in field Op3 matches the ratio of the values in Op1 to Op2. For uncompressed images, this test always returns TRUE (with a message indicating that these tests are not performed on uncompressed images). For divide by zero, this test returns FALSE.	Boolean	Op1: EntityField or EntityElement Op2: EntityField or EntityElement Op3: EntityField or EntityElement	
Result	**If/Then Result**	Conditional Test evaluates the conditional statement of IF Op1, THEN Op2. The expression returns Op2 if Op1 is TRUE and PASS otherwise.	Test Result	Op1: Boolean Op2: TestResult	IfThenResult(Op1, Op2)
	If/Then/Else Result	Conditional Test evaluates the conditional statement of IF Op1, THEN Op2, ELSE Op3. The expression returns Op2 if Op1 is TRUE and Op3 otherwise. If Op2 or Op3 is of type Boolean, the Expression returns PASS for TRUE and FAIL for FALSE values.	Test Result	Op1: Boolean Op2: TestResult or Boolean Op3: TestResult or Boolean At least one of Op2 and Op3 must be a Test Result	IfThenElseResult(Op1, Op2, Op3)
	Generate Test Result	Returns a TestResult that is provided as the operand Op1. If Op1 is Boolean, then True is converted to Pass and False is Converted to Fail.	Test Result	Op1: Boolean or TestResult	ReturnResult(Op1)
Numeric	**Numeric Value**	Returns the Numeric Value represented by the operand Op1. If Op1 is a String, it must be a properly formatted number with no leading zeros. If Op1 is a Byte Set, the entire set of bytes is interpreted as a value. Ex: NV(Set-Byte([0x01, 0x00])) is 256	Numeric	Op1: String or Byte Set	NV(Op1)
	Count	Returns the number of values found in the Set operand Op1. This is frequently used with Entity Sets to find the occurrence of an entity, for	NumericInteger	Op1: Set (Any Type)	Count(Op1)

Expression Definitions				
Expression Name	**Description**	**Return Type**	**Operand Types**	**Operator Syntax**
	example: Count(Recs(Rec(10)) provides the number of Record Type-10. Also used to count the number of characters or bytes: Count(Chars({Fld(1.001)})) provides the number of characters in the field. Count(B{Rec(1)}) provides the number of raw bytes in the record.			
Record Type	Returns the Record Type of the operand Op1.	Numeric	Op1: EntityRecord, EntityField, EntitySubField, EntityInfoItem, or EntityElement	RecType(Op1)
Field Number	Returns the field number of the operand Op1.	Numeric	Op1: EntityField, EntitySubField, EntityInfoItem, or EntityElement	FieldNum(Op1)
Modulo	Returns the remainder of the Op1 divided by Op2.	Numeric	Op1: NumericInteger Op2: NumericInteger	Mod(Op1, Op2)
Add	Returns the sum of the two operands Op1 and Op2.	Numeric	Op1: Numeric Op2: Numeric	Add(Op1, Op2)
Subtract	Returns the difference of the two operands Op1 and Op2.	Numeric	Op1: Numeric Op2: Numeric	Sub(Op1, Op2)
Multiply	Returns the product of the two operands Op1 and Op2.	Numeric	Op1: Numeric Op2: Numeric	Mult(Op1, Op2)
Divide	Returns the quotient of the two operands Op1 and Op2.	Numeric	Op1: Numeric Op2: Numeric	Div(Op1, Op2)
Minimum	Returns the minimum numeric value in the set Op1.	Numeric	Op1: Set-Numeric	Min(Op1)
Maximum	Returns the maximum numeric value in the set Op2.	Numeric	Op1: Set-Numeric	Max(Op1)

Expression Definitions					
	Expression Name	**Description**	**Return Type**	**Operand Types**	**Operator Syntax**
String	**String Value**	Returns the String Value generated by decoding the binary data of the Entity Operand Op1 according to the specified character encoding for that Entity. For EntityElements, leading and trailing whitespace is ignored. Use WS{Op1} to force leading and trailing whitespace to be included for EntityElements. This is important because XML Elements frequently contain leading and trailing whitespace e that is not part of the data, but rather for formatting.	String	Op1: EntityField, EntityInfoItem, or EntityElement	{Op1}
	String Value (With Leading and Trailing Whitespace)	Returns the String Value (including leading and trailing whitespace) generated by decoding the binary data of the EntityElement Operand Op1 according to the specified character encoding for that Entity.	String	Op1: EntityElement	WS{Op1}
	Element Name	Returns the String Value that represents the XML Element name.	String	Op1: EntityElement or EntityContainerElement	ElmName(Op1)
	String from ASCII	Returns the String Value of the ASCII code operand Op1. If any value in the operand is outside the range of 7-bit ASCII values, this expression returns an empty String. For Set-Numeric operands, each number in the set represents one ASCII character. Ex. ASCII(Set-Num([0x30, 0x31])) is 01 and ASCII(Num(33)) is !	String	Op1: Numeric or Set-Numeric	ASCII(Op1)
	String from Unicode	Returns the String Value of the Unicode Codepoint operand Op1. If any codepoint in the operand is invalid, it returns an empty String. For String sets, each string represents one Unicode codepoint (and thus one character) Ex. Unicode(Set-Str([U+0030, U+0033])) is 03	String	Op1: String or String Set	Unicode(Op1)
Set	**Byte Values**	Returns a Set of Byte Values representing the exact binary data contained in the Entity Operand Op1 in Big-Endian format. Note: this includes any separator characters. If Op1 is an EntityField, the field number FN (e.g. “1.001:”) is not part of the byte data. However, if Op1 is an EntityRecord, all data in the record,	Set-NumericByte	Op1: EntityField, EntityInfoItem, EntitySubField, EntityRecord	Bytes(Op1)

Expression Definitions				
Expression Name	**Description**	**Return Type**	**Operand Types**	**Operator Syntax**
	including field numbers, is part of the data.			
Character Values	Returns a Set of String Values, with each string representing one character in the operand Op1. Ex: Chars(Str(value)) is Set-Str([v,a,l,u,e])	Set-String	Op1: String	Chars(Op1)
List Values	Returns the Set of String Values found in the StringList Op1. Ex: ListValues(StrList(A|B|C)) is Set-Str([A,B,C])	Set-String	Op1: StringList	ListValues(Op1)
String Values	Returns the set of String Values generated by decoding the binary data of each of the Entities in the Entity-Set Operand Op1 according to the specified character encoding for that Entity. For EntityElements, leading and trailing whitespace is ignored. Use WS{Op1} to force leading and trailing whitespace to be included for EntityElements. This is important because XML Elements frequently contain leading and trailing whitespace e that is not part of the data, but rather for formatting.	Set-String	Op1: Set-Entity	Set-{Op1}
Numeric Values	Returns the set of Numeric Values represented by the set of Strings in Op1, which must be properly formatted numbers with no leading zeros.	Set-Numeric	Op1: Set-String	Set-NV(Op1)
Binary from Base64	Returns a set of bytes that represents the decoded Base-64 value found in Op1. If Op1 is not a valid Base-64 encoded string, this expression returns an empty set.	Set-NumericByte	Op1: String Op1 must be a valid Base-64 string	B64toBytes(Op1)
Record Set	Returns the Set of Records present in the transaction. If Op1 is specified, it represents the record type desired. An Integer value represents the Record Type. A Set-NumericInteger represents any number of Record Types, and a string represents the XML Element name of the record types desired. For example: Recs(Int(10)) returns the Set of Type-10 Records present in the transaction. Recs(Set-Int([13,14])) is the Set of Type-13 and Type-14 Records.	Set-EntityRecord	Op1: NumericInteger, Set-NumericInteger, or String	Recs(Op1)

Expression Definitions				
Expression Name	**Description**	**Return Type**	**Operand Types**	**Operator Syntax**
	Recs represents all records in the transaction.			
Field Set In Transaction	Returns the Set of Fields in the Transaction with the specified Field Number, that match the EntityField, or that exist in the EntityRecord indicated by operand Op1. For example: FldsInTx(Int(999)) is the Set of Fields 999 in any Record Type in the Transaction. FldsInTx(Fld(13.002)) is the set of Field 002 in any Type-13 Record. FldsInTx(Rec(10)) is the set of Fields in every Record Type-10. FldsInTx is the Set of all Fields in the Transaction. ***Note: This is the set of Fields among separate Records in the Transaction. For Fields in a single Record instance use FldsInRec.***	Set-EntityField	Op1: NumericInteger, EntityField, or EntityRecord	FldsInTx(Op1)
Field Set In Record	Returns the Set of Fields in a given Record operand Op1. If Op2 is specified it is the Field Number desired. For example: FldsInRec(Rec(1)) is the Set of Fields in Record Type-1. FldsInRec(Rec(10),Int(1)) is the set of Fields 001 in a single Record Type-10. ***Note: This is the set of Fields in one instance of a record. For Fields among several separate Records, use FldsInTx.***	Set-EntityField	Op1: EntityRecord Op2: NumericInteger	FldsInRec(Op1, Op2)
Subfield Set	Returns the Set of SubFields found in the Field operand Op1.	Set-EntitySubfield	Op1: EntityField	SubFldsIn(Op1)
Information Item Set In Transaction	Returns the Set of Information Items in the Transaction. If Op1 is specified, it represents the Information Item mnemonic desired. For example: InfoItemsInTx(Str(IDC)) is the Set of Information Items with mnemonic "IDC". InfoItemsInTx is the Set of all information items in the transaction. This expression identifies information items using their information separator tags, so even empty information items are returned by this expression. ***Note: This is the set of Information Items among separate Records in the Transaction. For Information Items in***	Set-EntityInfoItem	Op1: EntityField or EntitySubField	InfoItemsInTx(Op1)

Expression Definitions					
	Expression Name	**Description**	**Return Type**	**Operand Types**	**Operator Syntax**
		a single Record instance use InfoItemsIn.			
	Information Item Set In Record	Returns the Set of Information Items found in the Field or Subfield operand Op1. If Op2 is specified, it represents the Information Item mnemonic desired. For example: InfoItemsIn(Fld(1.003), Str(IDC)) is the Set of Information Items with mnemonic "IDC" in field 1.003. This expression identifies information items using their information separator tags, so even empty information items are returned by this expression. ***Note: This is the set of Information Items in one instance of a record. For Information Items among several separate Records, use InfoItemsInTx.***	Set-EntityInfoItem	Op1: EntityField or EntitySubField Op2*:* String	InfoItemsIn(Op1, Op2)
	Element Set in Transaction	Returns the Set of XML elements in every Op1 in the transaction. If Op2 is used, it provides the name of the desired elements. If ChElmsInTx is used, only direct child elements are considered. ElmsInTx with no operands returns the set of all elements in the transaction. ***Note: This is the set of elements among all Op1 instances in the Transaction. For elements in a single Op1 instance use ElmsIn.***	Set-EntityElement or SetEntityContainerElement	Op1: EntityElement or EntityContainerElement Op2: EntityElement or EntityContainerElement	ElmsInTx(Op1, Op2) Or ChElmsInTx(Op1,Op2)
	Element Set	Returns the Set of XML elements in a specific Op1. If Op2 is used, it provides the name of the desired elements. If ChElmsIn is used, only direct child elements are considered. ***Note: This is the set of elements in one instance of an Op1. For elements among several instances of Op1, use ElmsInTx.***	Set-EntityElement or SetEntityContainerElement	Op1: EntityElement or EntityContainerElement Op2: EntityElement or EntityContainerElement	ElmsIn(Op1, Op2) Or ChElmsIn(Op1,Op2)Op1,Op2)
Generic	**Select From Set**	Returns the Value found at the specified 1-based index in the Set Op1. Example: Select(Int(2),Set-Str([A,B,C,D])) will return Str(B), the second String in the set.	Any	Op1: NumericInteger Op2: Set (Any Type)	Select(Op1, Op2)
	Select SubSet	Returns the subset of Set Op2 indicated by the 1-based index values represented by Set-Int Op1. Invalid indices are ingored. Example:	Set-Any	Op1: Set-Int Op2: Set (Any Type)	SelectSubSet(Op1, Op2)

Expression Definitions				
Expression Name	**Description**	**Return Type**	**Operand Types**	**Operator Syntax**
	SelectSubSet(Set-Int[1,3,4], Set-Str([A,B,C])) will return Set-Str([A,C]) (index 4 is ignored)			
First Occurrence	Returns the First Value in the Set Op1.	Any	Op1: Set (Any Type)	FirstIn(Op1)
Last Occurrence	Returns the Last Value in the SetOp2.	Any	Op1: Set (Any Type)	LastIn(Op1)
Set Union	Returns the Set Union of the two Set operands Op1 and Op2 that contains all of the members of both. Ex: Union(Set-Num([1.0,2.2]),Set-Num([2.2, 10, 200.1])) is Set-Num([1.0, 2.2, 10, 200.1])	Set (Any Type)	Op1: Set (Any Type) Op2: Set (Any Type) Op1 and Op2 must be of same set type.	Union(Op1, Op2)

Complex Test Assertions

Some Test Assertions cannot be represented using the well-defined Test Assertion Syntax, or require specific instances of entities to be identified rather than every occurrence as defined by the Entity Value-Types. Such Assertions require a textual description of the test that must be performed. To assist in streamlining most of these textual descriptions, a Complex Assertion Syntax is included in this section. This Complex Assertion Syntax does not represent a complete syntax, nor is it necessarily well-defined; its purpose is only to provide a toolset to assist in explaining Complex Test Assertions. The Complex Assertion Syntax is composed of Complex Expressions, Complex Value-Types, and Complex Procedures. Complex Procedures are operations that are repeated for several Assertions, but do not return a value. The format is: Complex(Description), where Description is composed of Complex Expressions, Complex Value-Types, Complex Procedures, and plain English. Description may also be a message such as "See Note" when the test assertion is too large to be contained in the table, and must be explained in a note or other location.

Complex Expressions and Value-Types

The Expressions found in this section are used to help clarify Complex Assertions that cannot easily be represented in the Assertion Syntax.

Table 5.6 - Assertion Syntax: Complex Expression Definitions

Complex Expression Definitions				
Expression Name	**Description**	**Return Type**	**Operand Types**	**Operator Syntax**
Parent	Returns the Parent of the Entity (the Entity that contains the specified Entity).	GenericEntity	Op1: GenericEntity	Parent(Op1)

Pair	Returns a set of Entity Pairs that satisfy the Entity Query. The pair is represented in the rest of the Assertion by A,B.	Set (GenericEntity)	Op1: EntityQuery	Pair(Op1)

Table 5.7 - Assertion Syntax: Complex Value-Type Definitions

Complex Value-Type Definitions		
Value Type	**Valid Values**	**Syntax**
GenericEntity	Any generic entity defined for either encoding: Transaction, Record, Field, Subfield, InfoItem, or Element. This is a generic representation that does not indicate a specific entity.	GenEntity(VALUE)
EntityQuery	A search query for a specific instance of an Entity. The general structure of the query is: P:N in Q ST(condition) For example, InfoItem:2 in Subfield ST(EQ({InfoItem}, Str(1)) Note that any of the elements of the query are optional, for example: P:N or P in Q or P ST(condition) or simply P	Query(VALUE)

Complex Procedures

The procedures described in this section list common tasks that are repeated for several Complex Assertions. They are not Expressions because they do not return a value.

Table 5.8 - Assertion Syntax: Complex Procedure Definitions

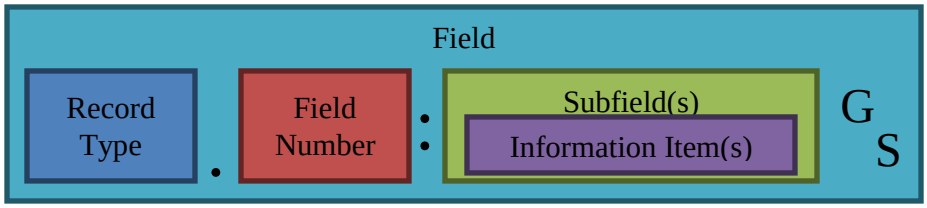
Complex Procedure Definitions				
Name	**Description**	**Return Type**	**Operand Types**	**Operator Syntax**
Variable	Uses the String value as a variable to represent the containing expression for the remainder of the Assertion Text. This is used to avoid repetitions in the assertion text.	None	Op1: String	Var(Op1)(Expression)
For Loop	Evaluates the containing expressions for each value in the specified range of the operator values (Op1 to Op2). X represents the current value in the loop.	None	Op1: NumericInteger Op2: NumericInteger	ForX(Op1, Op2) (Expressions)
For Each Loop	The For Each Loop evaluates the containing expressions for each value in the set Operator. X represents the current value in the loop. For nested loops, XN represents the current value of the loop, where N is the level of nesting.	None	Op1: Set (Any Type)	ForEachX(Op1) (Expressions)
Next Iteration	Only for use in For Each Loop: references the next occurrence of X.	None	None	Next(X)
Previous Iteration	Only for use in For Each Loop: references the previous occurrence of X.	None	None	Previous(X)

Field Definitions and Structures (Traditional Encoding)

The test assertion syntax represents all field types as a field that contains a list of one or more subfields, each of which contains a list of one or more information items. Fig. 5.9 is a representation of how each field type is represented by the test assertion syntax:

- Single Information Item: Field with one subfield containing one information item.
- Multiple Information Items: Field with one subfield containing multiple information items.
- Subfields Repeating Sets of Information Items: Field with one or more subfields, each containing sets of one or more information items.
- Subfields Repeating Values: Field with one or more subfields, each containing one information item.

Figure 5.9 - Generic AN-2013 Field Structure



Unless otherwise stated, the Test Assertion Syntax expresses all field structures using the Traditional notation of record type and field number (e.g., 1.001) as well as subfield and information item indices when appropriate. However, the NIEM-XML encoding has no concept of subfields or information items. Instead, the XML encoding uses sub elements. Annex G of the AN-2013 standard - NIEM-conformant Encoding Rules - can be used to translate the listed values for Traditional structures to the XML equivalent. In some cases the tables of requirements and assertions list the XML element names when necessary for clarifying an assertion.

Tables of requirements and assertions format

Table Layout

A Requirement is related to one or more Test Assertions. Therefore, a single Requirement may require more than one test assertion row of a table. To represent this association (from left to right), a Requirement is listed first, followed by the related assertion(s). Test notes may be included for any specific assertion to help clarify its meaning when necessary. Information contained in the table is described by the table column headers later in this section.

The AN-2013 contains requirements for several biometric record types, data conventions, and data encodings contained in various clauses, tables, figures, and annexes throughout the standard. The complex and detailed nature of the AN-2013 standard, including the variety of ways that requirements are specified, increases the chance that certain requirements may be stated in more than one section of the standard. In such cases, the table of requirements and assertions would indicate that the requirement is a duplicate requirement. The AN-2013 section numbers where the requirement is specified would be indicated. If there are too many to list, the requirement would be labeled a generic requirement, where other sections define it with more clarity and detail. For all duplicate or generic requirements, no assertions would be defined. The columns of the table

dedicated to assertion information may be merged and filled with a notification of the duplicate requirement. The format would be as follows, where Optional Message is any text that helps describe why the requirement is a duplicate or generic:

- Duplicate Requirement. Optional Message. See AN-2013 Section:
- Generic Requirement. Optional Message. Specific instances are defined more precisely in several sections of AN-2013.

For an example of the table layout specified by the CTMF, see Annex B.

Table Headers

The following describe the headings for the tables of requirements and assertions format:

- **Requirement # and ID:** Includes a unique AN-2013 requirement number and a unique identifier for the requirement and associated assertion or set of assertions. For Record Type requirements, the Requirement # is in the form RTN.M, where N is the Record Type and M is the sequential number of the requirement (for requirements in the annexes the form is AN followed the Annex letter). For sections not associated with a record type and annexes, the prefix is SEC followed by the section number. If additional requirements must be entered in the future, the number M may change. The Requirement ID provides reference to the type of requirement (e.g., transaction, record, or field), and is in the form of "Type: Description" where type may be "Transaction", "Record", or "Field". For requirements found in Annex B of the AN-2013 standard, the Requirement ID is preceded by "Traditional-". For requirements found in Annexes C and G of the AN-2013 standard, the Requirement ID is preceded by "NIEM-".
- **Ref. in Base Std. (Reference in Base Standard):** Identifies the clause (or section) where the requirement is included in the AN-2013 standard. In some cases the reference includes additional information such as a Table number.
- **Requirement Summary:** Provides a summary of the requirement detailed as textual information or an interpretation of the requirement in the standard. It provides the essentials of the requirement but may not provide all the text necessary to understand it.
- **Level:** Indicates whether Level 1 or Level 2 conformance testing is required to address the assertion identified in the Assertion ID column of the same row. Level 3 conformance tests are indicated only when necessary to show that the requirement is not currently testable or addressed.
- **Assertion ID:** Provides an identifier of a specific test assertion within the set of test assertions associated with a requirement.
- **Test Assertion:** Provides, whenever possible, a mathematical equation or a procedure using the language specified by the Assertion Syntax.
- **Notes:** Contains the ID of the test note, in the form t##. Test notes provide additional information related to the assertion and are included below the tables.
- **Imp. Required (Implementation Required):** The Imp. Required column indicates whether or not the assertion must be supported in the Imp. Support column. The format is CondCode-Entity, where:

  Entity:

This indicates the entity (Field, Subfield, etc.) that is the primary subject of the assertion. The assertion must be tested for every instance of this entity in the transaction. This is particularly useful for assertions that contain more than one entity (generally Level-2 assertions). The syntax is any Entity type in Table 5.1 (Assertion Syntax: Value-Type Definitions Value-Types) defined in the Assertion Syntax.

Cond Code:

This indicates the Cond Code (as specified in AN-2013) of the Entity that is the subject of the assertion. The Cond Code indicates whether or not the entity must be present. For XML Elements that do not relate exactly one-to-one with Traditional constructs, the Cond Code is M (Mandatory) if the Cardinality is greater than 0 and O (Optional) otherwise. All assertions associated with entities that have Mandatory Cond Codes must be claimed under Implementation Support. Note that the Cond Code of an entity is dependent upon the inclusion of the parent record or field in the IUT. For example, Field 10.001 with Cond Code M is required to be present (and therefore its related assertions are required to be claimed) only if a Type-10 Record is included. As another example, 1.013-DNM is mandatory only if the optional field 1.013 is included. It should also be noted that the Cond Code only applies if the assertion is related to the transaction's encoding as indicated by the Enc. (Encoding) column. The Cond Code values are:

- **M:** Mandatory – entity must be present, assertion must be claimed
- **O:** Optional – entity not required to be present, assertion not required to be claimed
- **D:** Dependent – presence of entity is dependent upon certain conditions specified in the AN-2013 standard. If the entity is required to be present, the assertion must be claimed.
- **M⇑:** Mandatory within the optional field/subfield – entity must be present if the containing field/subfield is present, and the assertion must then be claimed.
- **O⇑:** Optional within the optional field/subfield – entity is not required to be present, even if the containing field/subfield is present. The assertion is not required to be claimed.

Note: For Optional or Dependent Cond Codes, if the entity is present in the IUT (although it is not required to be), the related assertion must be claimed.

Example:
M-Fld(10.001). The subject of the assertion is Field 10.001 with Cond Code M, meaning that the assertion must be tested for every instance of Field 10.001. This is a Mandatory field (M), indicating that this assertion must be claimed given that Record Type-10 is present in the IUT.

- **Imp. Support (Implementation Support):** Denotes a supplier's implementation support of a particular assertion ("Y"/"N"). A note can follow the table when providing more details of implementation support (or the lack of it) is required. For assertions with Mandatory Imp. Required values, the Imp. Support should be Y given that the parent Record or Field is also supported.
- **Supported Range:** Indicates a range of values supported, especially when it is different than the full range of values specified in the standard. When an information item is specified as a single value, or does not address a range of values, a N/A should be used.
- **Test Result:** This column is used to denote the test results. The result is one of "Pass", "Fail", or "Warning". Explanatory notes can be added below the table, for example when a

Warning is given. The result is the value provided by evaluating the test described in the Test Assertion column.

- **Enc. – (Encoding):** This table header indicates which assertions differ (in values required or conditions) between Traditional and NIEM encoding. This table header does not indicate which assertions are addressed by the XML Schema and which will need to be addressed in code. Valid values are:
    - T: The assertion only applies to the Traditional encoding as described in Annex B of AN-2013.
    - X: The assertion only applies to the NIEM-conformant (XML) encoding as described in Annex C of AN-2013.
    - B: The assertion is applicable to both Traditional and NIEM (XML) encoding.
        - Following the conventions in the AN-2013 standard, test Assertions are expressed using constructs (fields, records, etc.) found in Traditional encoding (such as 1.002). The same assertion applies for the XML elements that correspond to the Traditional constructs. For example, 10.006 in Traditional Encoding corresponds to XML Element <biom:ImageHorizontalLineLengthPixelQuantity>. Annex G of the AN-2013 provides a mapping between Traditional and XML encodings.

5.5 Claim of Supported Test Assertions

The table format for requirements and assertions provides the means for the developers of implementations under test (IUT) to claim in the tables the list of all the assertions supported.

This information is useful to the IUT supplier as a checklist on the content of their implementations and also useful to testing laboratories that would evaluate conformance of these IUTs against the supplier's claims. Two columns in the tables are included to provide this information: Implementation Support column (YES/NO/Partial) and Supported Range column (if Implementation Support is "Partial", the supported range should be provided).

The minimum implementation requirements are documented in Annex A. The Implementation Required (Imp. Required) column indicates the entity related to each assertion, and the Cond Code for that entity. A Mandatory Cond Code indicates that the entity must be present, and therefore the assertion must be claimed. Such Cond Codes only apply if the containing Record or Field is also present—for example, 10.001 is Mandatory, but only if the IUT contains a Record Type-10. Regardless of the supplier claims, if an entity (Field, Subfield, etc.) is included in the IUT, the test assertions related to that entity will be tested and should be reported by a conformance test tool.

It is recommended that if the IUTs are sent to a testing laboratory, the IUT provider submit the information below to the laboratory:

- Provider name
- Provider address
- Transaction identifier
- Transaction version number
- Additional implementation information (optional)
- Submission date

- For each claimed Record Type, provide the Record Type number and whether or not (Yes or No) there are any known deviations from (or exceptions to) the requirements found in the base standard and identified in the Conformance Testing Methodology for the associated Record Types in the IUT. For specific exceptions, the Implementation Support column of the tables of requirements and assertions can be used to indicate the difference on a per-assertion basis. In addition, if the deviation is general and applies to the entire Record Type, a description should be provided. This option is useful for cases where there have been modifications to the base standard that are not reflected in the conformance testing methodology, where the IUT provider believes there is a defect in the base standard or conformance testing methodology, and other instances where the implementation does not fully conform to the AN-2013 standard requirements.

The testing laboratory may use testing tools that implement this CTMF and any the test assertions included in any derivative conformant publications (such as those which document additional requirements) to provide a determination of the level of conformance of the IUT to the AN-2013 standard.

Annex A: Minimum Support for AN-2013 Record Types and Interrelated Fields

A.1 Minimum Conformance

This document includes conformance test assertions for all the transaction-related requirements specified in AN-2013. This document does not include test assertions for Record Types other than Record Type 1.

This section identifies the AN-2013 requirements and the conformance test assertions that are required for every transaction according to the terms specified in the AN-2013 standard. At a minimum, AN-2013 requires that:

- the transaction adheres to its specified encoding (Traditional or NIEM-XML) requirements
- the transaction includes one and only one Record Type-1
- Record Type-1 is encoded exclusively in 7-bit ASCII (for Traditional Encoding)
- Record Type-1 is conformant to the requirements specified for its fields, subfields, and information items.
    - All mandatory fields, subfields, and information items in Record Type-1 must be present (with data), and the requirements for those entities must be met.
    - Optional and dependent fields, subfields, and information items that are present in Record Type-1 must be conformant to the requirements specified for those entities.
- the transaction includes at least one other record of a type other than Record Type-1
- the transaction does not include deprecated or reserved record types or fields

The AN-2013 requirements listed above constitute what is indicated in this document as the minimal conformance for any AN-2013 transaction, according to the requirements specified by AN-2013. Note, however, that minimal conformance is silent regarding the requirements for individual Records other than Record Type-1. For example, a minimally conformant AN-2013 transaction may include a non-conformant Record Type-10.

A.2 Interrelated Field Support

Section 7: Information Common to Several Record Types in the AN-2013 standard includes requirements for fields that are common among various record types. These requirements are contained in Annex C, in the table of requirements and assertions associated with Section 7 of the AN-2013 standard. Below is a list of the common fields with requirements specified in Annex C of this CTMF.

Table A.1 - AN-2013 Interrelated Field Support

Support for AN-2013 Interrelated Fields		
Number	**Field Contents**	**Support**
xx.001	Record header	All Record Types. See Field: xx.001-Record Header
xx.002	Information designation character / IDC	All Record Types except Record Type-1. See Field: xx.002-IDC
xx.995	Associated Context / ASC	Record Types 10 and above, not including 21 and 98. See Field: xx.995-ASC through Field: xx.995-ASC-ASP
xx.997	Source Representation / SOR	Record Types 10 and above, not including 18, 21, and 98. See Field: xx.997-SOR through Field: xx.997-SOR-RSP
xx.016	Segments / SEG	Record Types 20 and 21. See Field: xx.997-SOR-RSP and Field: xx.995-ASC-ASP
xx.021	SRN, ACN	Record Types 20 and 21. See Field: xx.997-SOR-SRN and Field: xx.995-ASC-ACN

Annex B: Sample Requirement and Assertion Table Format

This section describes the layout of the table-based requirements and assertions format required by the CTMF.

Figure B.1 – Sample Requirements and Assertions Table

Req. # - ID	Red. In Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
RTX.1 – Requirement ID	N.N, Table Z	Requirement text: verbatim from the base standard or a summary.	1	Assertion ID 1	Assertion summary written using the Test Assertion Syntax – this one is for Traditional encoding		M-Fld(1.001)			Warning	T
			1	Assertion ID 2	Assertion summary written using the Test Assertion Syntax – this one is for XML encoding	t##	O-				X
			2	Assertion ID N	Assertion summary written using the Test Assertion Syntax – one or more assertions are listed per Requirement						B
RTX.2– Requirement ID2	M.M	Requirement text: verbatim from the base standard or a summary.	1	Assertion ID 1	Assertion summary written using the Test Assertion Syntax – this one is for both encodings						B

Annex C: Tables of Requirements and Assertions

The CTMF allows for all requirements and assertions to be documented for AN-2013. The full range of requirements and assertions are not included in this publication. The requirements necessary for minimum conformance as indicated in Annex A are included as are requirements for data formats, encodings, and those related to several record types. This Annex lists the tables of requirements and assertions common to most AN-2013 transactions; the AN-2013 requirement types and the reason for their inclusion are provided below:

AN-2013 Section 5: Data Conventions
Requirements for data conventions describe the structure and ordering of constructs that make up all AN-2013 transactions. Requirements for deprecated Record Types 3, 5, and 6 are also included to check for nonexistence of these Record Types. Additionally an assertion is specified that checks for the nonexistence of reserved Record Types 22 through 97.

AN-2013 Section 7: Information Common to Several Record Types
Information common to several record types refers to fields and other constructs that are defined once in the AN-2013 standard, and repeated for several record types. While these requirements may not be relevant to every transaction, a portion of these requirements are relevant to a large number of transactions.

AN-2013 Section 8.1 Record Type-1: Transaction information record
One and only one instance of Record Type-1 is required to be included in every AN-2013 transaction. The requirements associated with mandatory fields, subfields, information items, and XML Elements in Record Type-1 must be met for every AN-2013 transaction. Note that if any optional construct is present in any transaction, the defined requirements for those constructs are mandatory for conformance. An optional construct does not indicate an optional requirement.

AN-2013 Annex B: Traditional Encoding
Traditional encoding specifies requirements that describe the general makeup of any traditionally-encoded AN-2013 transaction.

AN-2013 Annex C: NIEM Conformant encoding
NIEM conformant encoding specifies requirements that describe the general makeup of any XML-encoded AN-2013 transaction.

AN-2013 Annex G: Mapping to the NIEM IEPD
Mapping to the NIEM IEPD provides the information necessary to interpret requirements represented in Traditional Encoding notation for XML-encoded transactions. The mapping indicates instances where the relationship between Traditional constructs (fields, subfields, etc.) is not one-to-one with XML elements.

Table C.1 - Assertions for Transaction-related Requirements

Req. # - ID	Ref.in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Transaction-related Requirements											
TX.1 - Transaction: Required Record Types	5.1, 5.3.1, Annex G	There may be multiple records in a transaction of each record type other than Type-1. The only required record Type is Type-1, which is used to describe the transaction. There shall be at least one other record type from Table 3 accompanying a Record Type-1. Transmissions to be exchanged are required to contain one and only one Type-1 record per transaction. Itl:PackageInformationRecord Cardinality 1..1	1	Transaction-Type-1-Required	EQ (Count(Recs(Int(1))), Int(1))		M-Transaction				T
			1	NIEM-Transaction-PackageInformationRecord-Required	EQ (Count(Recs(Str(itl:PackageInformationRecord))), Int(1))		M-Transaction				X
			1	Transaction-Required-Additional-Record	GT (Count(Recs), Int(1))		M-Transaction				B
TX.2 - Transaction: Single Subject	5.1	All records in a transaction shall pertain to a single subject. Biometric data used to identify another individual requires a separate transaction.	3	Transaction-Single Subject	ReturnResult (Result (Warning(Unchecked Level 3 – All records shall pertain to a single subject.)))		M-Transaction				B
TX.3 - Transaction: Records Transmitted Together	5.1	All of the records belonging to a single transaction shall be transmitted together.	3	Transaction-Records Together	ReturnResult (Result (Warning(Unchecked Level 3 – All records belonging to a single transaction shall be transmitted together.)))		M-Transaction				B
TX.4 - Transaction: Size	5.2	The upper limit of 1000 records is maintained in this version of the standard to ensure backward compatibility with the	1	Transaction-Size	LTE (Count(Recs),		M-Transaction				B

Req. # - ID	Ref.in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Transaction-related Requirements											
		2007 version.			Int(1000))						
TX.5 - Transaction: Reserved Records	5.3 Table 3	22-97 reserved for future use.	1	Transaction-Records Reserved	EQ (Count(Recs(Set-Int(22 to 97))), Int(0))		M-Transaction				T
			1	NIEM-Transaction-Records Reserved	ReturnResult (Result (Pass(Element names are not defined for reserved records. Invalid records will fail schema validation.)))		M-Transaction				X
TX.6 - Transaction: Type1-Record_First	5.3.1	The Type-1 record shall always be the first record within the transaction.	1	Transaction-Type1-First	EQ (RecType(FirstIn(Recs)), Int(1))		M-Transaction				T
			1	NIEM-Transaction-Type1-First	EQ (ElmName(FirstIn(Recs)), Str(itl:PackageInformationRecord))		M-Transaction				X
TX.7 - Transaction: Type3-Deprecated	5.3.3, 5.4, Table 3., 8.3	Record Type-3 shall not be contained in transactions conforming to this version of the standard. No instances of Record Type-3 shall be included in a transaction conformant with this version of the standard. Deprecated records for this version are Record Types 3, 5 and 6.	1	Transaction-Type3-Zero Occurrences	EQ(Count(Recs(Int(3)), Int(0))		M-Transaction				T
			1	NIEM-Transaction-Type3-Zero Occurrences	ReturnResult (Result (Pass(Deprecated Records are not defined for NIEM-XML; invalid Records will fail Schema validation)))		M-Transaction				X

Req. # - ID	Ref.in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Transaction-related Requirements											
TX.8 - Transaction: Type5-Deprecated	5.3.5, 5.4, Table 3, 8.5	Record Type-5 shall not be contained in transactions conforming to this version of the standard. No instances of Record Type-5 shall be included in a transaction conformant with this version of the standard. Deprecated records for this version are Record Types 3, 5 and 6.	1	Transaction-Type5-Zero Occurrences	EQ (Count(Recs(Int(5))), Int(0))		M-Transaction				T
			1	NIEM-Transaction-Type5-Zero Occurrences	ReturnResult (Result (Pass (Deprecated Records are not defined for NIEM-XML, but invalid Records will fail Schema validation)))		M-Transaction				X
TX.9 - Transaction: Type6-Deprecated	5.3.6, 5.4, Table 3, 8.6	Record Type-6 shall not be contained in transactions conforming to this version of the standard. No instances of Record Type-6 shall be included in a transaction conformant with this version of the standard. Deprecated records for this version are Record Types 3, 5 and 6.	1	Transaction-Type6-Zero Occurrences	EQ (Count(Recs(Int(6)), Int(0))		M-Transaction				T
			1	NIEM-Transaction-Type6-Zero Occurrences	ReturnResult (Result (Pass (Deprecated Records are not defined for NIEM-XML, but invalid Records will fail Schema validation)))		M-Transaction				X
TX.10 - Transaction: Reserved Character Types	5.5	The special characters “STX”, “ETX”, “FS”, “GS”, “RS”, and “US” are reserved and shall not be included in any data (except data marked as character type B).	1	Transaction-Reserved Character Types	ReturnResult (Result (Pass(Character Type assertions are performed on all data. These assertions test for the presence of reserved characters. Refer to the individual tests on data to determine the results.))		M-Transaction				T

Req. # - ID	Ref.in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Transaction-related Requirements											
					)						
TX.11 - Transaction: IDC Sequential	7.3.1	The value of the IDC shall be a sequentially assigned positive integer starting from zero and incremented by one up to a maximum of 99. IDC references are stated in Type-1 Field 1.003 Transaction content / CNT and shall be used to relate information items in the CNT field of the Type-1 record to the other records in the transaction.	2	Transaction-IDCSequentialValues	Complex (When arranged in numeric order, the set of all IDC values in the transaction must begin with 0, increment by 1, and the greatest value in the set must be equal to or less than 99)	t9	M-Transaction				T
TX.12 - Transaction: IDC Matching Values	7.3.1	Two or more records may share a single IDC solely to identify and link together records that pertain to different representations of the same biometric trait. Two or more image records may share a single IDC only when they are enhancements of a single image; such transformations shall have identical dimensions.	2	Transaction-MatchingIDCValues-ComparableBiometricTypes	Complex(See Note)	t2	M-Transaction				B

			2	Transaction-MatchingIDCSameImageDimension	Complex
					(
					ForEach(Pair(A,B) of Records with matching IDC fields)
					{
					
					{A.006} EQ {B.006}
					AND
					{A.007} EQ {B.007}
					
					}
					)
							M-Transaction				B

			3	Transaction-IDCsFromSameImage	ReturnResult (Result (Pass(Not feasible to test if the samples are from the same image, only that the samples come from the same type of biometric trait)))		M-Transaction				B
TX.13 - Field: 1.003-	8.1.3, Table 22,	IDC references are stated in Type-1 Field 1.003 Transaction content / CNT and shall	2	Transaction-CNT-	Complex (		M-Transaction				T

Req. # - ID	Ref.in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Transaction-related Requirements											
Transaction Content Subfield 2 IDC Matches	7.3.1	be used to relate information items in the CNT field of the Type-1 record to the other records in the transaction. It also specifies the order in which the remaining records shall appear in the file.		REC-IDC-Matches Records	ForEach (Record in Recs) (Present(Subfield in 1.003 ST AND (EQ({InfoI(1.003.REC) in Subfield}, RecType(Record)), EQ({InfoI(1.003.IDC) in Subfield}, {Record.002}))) Note: The record types must appear in the same order that they are listed in Fld(1.003))						
			2	NIEM-Transaction REC-CNT-IDC-Matches Records	Complex (ForEach (Record in Recs) (Present(XElm(1.003.biom:ContentRecordSummary) ST AND (EQ({XElm(1.003.biomRecordCategoryCode) in XElm(1.003.biom:ContentRecordSummary)}, RecType(Record)), EQ({XElm(1.003.ImageReferenceIdentification) in XElm(1.003.biom:ContentRecordSummary)}, {Record.002}))) Note: The record types must appear in the same order that they are listed in Fld(1.003))		M-Transaction				B
TX.14 - Transaction: SRN Sequential	7.3.2.1	The value of the SRN shall be a sequentially assigned positive integer starting from one and incremented by one, not to exceed 255.	2	Transaction-SRN-SequentialValues	Complex (When arranged in numeric order, the set of all SRN values in the transaction must begin with 1, increment by 1, and the greatest value in the set must be equal to or less than 255)	t9	M-Transaction				B

Req. # - ID	Ref.in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Transaction-related Requirements											
TX.15 - Transaction: ACN Sequential	7.3, 7.3.3.1	The value of the ACN shall be a sequentially assigned a positive integer starting from one and incremented by one, not to exceed 255.	2	Transaction-ACN-SequentialValues	Complex (When arranged in numeric order, the set of all ACN values in the transaction must begin with 1, increment by 1, and the greatest value in the set must be equal to or less than 255)	t9	M-Transaction				B

TX.16 - Transaction: T10 Matching	7.3, 7.3.4	There may be several Type-10 images of a particular part of the body. For instance, a photograph of a tattoo may cover the entire tattoo. Another may be a zoom-in shot of a portion of the tattoo. In order to link these two images, the same index number is assigned to Field 10.039: Type-10 reference number / T10, which is new to this version of the standard. Note that these images would have different IDC values.	2	Transaction-SameT10-DiffIDC	Complex
					(
					ForEach(Pair (A,B) of Records ST RecType(Records) EQ 10)
					{
					IF {A.039} EQ {B.039}
					THEN
					{A.002} NEQ {B.002}
					}
					)
							M-Transaction				T

			2	NIEM-Transaction-SameT10-DiffIDC	Complex
					(
					ForEach(Pair (A,B) of XElm(itl:PackageFacialAndSMTImageRecord)
					{
					IF {XElm(nc:IdentificationID) in XElm(biom:PhysicalFeatureReferenceIdentification) in A} EQ { XElm(nc:IdentificationID) in XElm(biom:PhysicalFeatureReferenceIdentification) in B}
					THEN
					{XElm(nc:IdentificationID) in XElm(biom:ImageReferenceIdentification) in A} NEQ { XElm(nc:IdentificationID) in XElm(biom:ImageReferenceIdentification) in B}
					}
					)
							M-Transaction				X

TX.17 - Transaction: Schema Validation	C.2, C.4, C.5.1	The ordering of elements is strict. The schemas referenced by this annex define the order and nesting structure of elements. The schemas also provide a W3C representation of the order and hierarchical	2	NIEM-Schema Validation	Complex (Perform and report Schema validation. Provide warning that the schema does not strictly enforce the standard, so the conformance of a	t10	M-Transaction				X

Req. # - ID	Ref.in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Transaction-related Requirements											
		structure of the XML content. The XML schema referenced for this encoding define the structure and order of the elements in the information exchange package. To the extent possible, the schema define data types and constraints that enforce the allowable content rules of the base standard. Nevertheless, the XML schema may not strictly enforce the allowable content. The base standard defines allowable content, and its requirements shall be met by implementers regardless of encoding method. All of this standard's required elements shall be present in a conforming instance document even if the schema referenced by this annex do not strictly enforce the requirement. The base standard defines allowable content, and its requirements shall be met by implementers regardless of encoding method.			transaction cannot be claimed from the result of schema validation. However, the Schema validation does indicate that structural requirements have been met, including appropriate ordering of the elements.)						
TX.18 - Transaction Valid Encoding	C.4.1	Each XML information element, tags and data content, shall be represented by a character set that is a subset of Unicode and that is allowable by W3C XML. Characters shall be transmitted using a Unicode encoding.	1	NIEM-XML Encoding	MO ({FirstIn(ElmsInTx)}, Set-Str([UTF-8, UTF-16, UTF-32]))		M-Transaction				X
TX.19 - Transaction Encoding Declaration	C.4.1	XML packages shall include an XML declaration that specifies the encoding.	1	NIEM-XML Declaration	EQ (ElmName(FirstIn(ElmsInTx)), Str(?xml))		M-Transaction				X
TX.20 - NIEM-Transaction Well-Formed	C.5.2, C.5.3	All separators are defined by the W3C XML recommendations. The characters "<" and ">" are reserved exclusively for enclosing XML element names. Every element with a start tag <Name> shall have	1	NIEM-Well-Formed XML	Complex (Test that the XML is well-formed according to W3C XML recommendations.		M-Transaction				X

Req. # - ID	Ref.in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Transaction-related Requirements											
XML		an end tag of format </Name>. For all logical records – including Types 4, 7, and 8 that do not have field tags in the Traditional encoding -- data elements are tagged according to XML rules. The format for each element shall consist of a start tag enclosed in angle brackets followed by data followed by an end tag.			)						

Table C.2 - Assertions for Record Type 1: Transaction information record

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
RT1.1 - Record: RecordHeaderFirst	7.1	The record header appears as the first field (xx.001) in each Record Type. The record header exists only in Traditional Encoding.	1	Type1-Field001First	EQ (FieldNum(FirstIn(FldsInRec(Rec(1)))), Int(1))		M-Rec(1)				T
			1	NIEM-Type1-Field001First	EQ (ElmName(FirstIn(ElmsIn(itl:PackageInformationRecord))), Str(biom:RecordCategoryCode))		M-Rec(1)				X
RT1.2 - Record: Type1-7-bitASCII	8.1, 5.6, Table 93	Note that since the alternate character encoding is specified in this record, there must be specified characters agreed upon in order to read this Record Type, particularly with Traditional encoding, and the characters that can be represented by the 7-bit ASCII code are those characters (see Table 93 for these characters).	1	Type1-ASCII	ReturnResult (Result (Pass(Character Type assertions are performed on all data in Record Type-1. These assertions are more restrictive because they test for character ranges that are a subset of 7-bit ASCII. Refer to the individual tests on data to determine the		M-Rec(1)				T

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
		Record Type-1 shall always be recorded in all encodings using the characters that can be represented by the 7-bit American National Standard Code for Information Interchange (ASCII) found in table 93 with the exception of the reserved values.			results.)))						
RT1.3 - Record: Type1-Reserved	Table 22	Table 22 specifies which fields are permitted to be present in a Type-1 Record. All others are reserved for future use.	1	Type1-ReservedFields	Not (AnyPresent(Set-Fld([1.019 to 1.999]))		M-Rec(1)				T
			1	NIEM-Type1-ReservedFields	ReturnResult (Result (Pass(Reserved Fields are not defined for NIEM-XML. The presence of any undefined elements will fail Schema validation)))	t10	M-Rec(1)				X
RT1.4 - Record: Type1-FieldOccurrence	Table 22, Annex G	Table 22 specifies the Field Occurrence for each field. Annex G specifies the cardinality for the XML elements.	1	Type1-1.001-Occurrences	EQ (Count(FldsInRec(Rec(1), Int(1))), Int(1))		M-Rec(1)				T
			1	Type1-1.002-Occurrences	EQ (Count(FldsInRec(Rec(1), Int(2))), Int(1))		M-Rec(1)				T
			1	Type1-1.003-Occurrences	EQ (Count(FldsInRec(Rec(1), Int(3))), Int(1))		M-Rec(1)				T
			1	Type1-1.004-Occurrences	EQ (Count(FldsInRec(Rec(1), Int(4))), Int(1))		M-Rec(1)				T

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
			1	Type1-1.005-Occurrences	EQ (Count(FldsInRec(Rec(1), Int(5))), Int(1))		M-Rec(1)				T
			1	Type1-1.006-Occurrences	MO (Count(FldsInRec(Rec(1), Int(6))), Set-Int([0,1]))		M-Rec(1)				T
			1	Type1-1.007-Occurrences	EQ (Count(FldsInRec(Rec(1), Int(7))), Int(1))		M-Rec(1)				T
			1	Type1-1.008-Occurrences	EQ (Count(FldsInRec(Rec(1), Int(8))), Int(1))		M-Rec(1)				T
			1	Type1-1.009-Occurrences	EQ (Count(FldsInRec(Rec(1), Int(9))), Int(1))		M-Rec(1)				T
			1	Type1-1.010-Occurrences	MO (Count(FldsInRec(Rec(1), Int(10))), Set-Int([0,1]))		M-Rec(1)				T
			1	Type1-1.011-Occurrences	EQ (Count(FldsInRec(Rec(1), Int(11))), Int(1))		M-Rec(1)				T
			1	Type1-1.012-Occurrences	EQ (Count(FldsInRec(Rec(1), Int(12))), Int(1))		M-Rec(1)				T
			1	Type1-1.013-	MO (		M-Rec(1)				T

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
				Occurrences	Count(FldsInRec(Rec(1), Int(13))), Set-Int([0,1]))						
			1	Type1-1.014-Occurrences	MO (Count(FldsInRec(Rec(1), Int(14))), Set-Int([0,1]))		M-Rec(1)				T
			1	Type1-1.015-Occurrences	MO (Count(FldsInRec(Rec(1), Int(15))), Set-Int([0,1]))		M-Rec(1)				T
			1	Type1-1.016-Occurrences	MO (Count(FldsInRec(Rec(1), Int(16))), Set-Int([0,1]))		M-Rec(1)				T
			1	Type1-1.017-Occurrences	MO (Count(FldsInRec(Rec(1), Int(17))), Set-Int([0,1]))		M-Rec(1)				T
			1	Type1-1.018-Occurrences	MO (Count(FldsInRec(Rec(1), Int(18))), Set-Int([0,1]))		M-Rec(1)				T
			2	NIEM-Type1-Cardinality	Complex(Check that all elements are in allowable cardinality ranges according to Annex G of the base standard. This may be achieved using Schema validation.)	t10	M-Rec(1)				X
RT1.5 - Field: 1.001-FieldStructure	Table 22, Annex B, Annex G	Table 22 specifies which fields contain subfields and information items as well as the number of occurrences permitted. A field contains a minimum of one subfield which contains a minimum of one information item.	1	1.001-SubfieldCount	EQ (Count(SubFldsIn(Fld(1.001))), Int(1))		M-Fld(1.001)				T
			1	1.001-InfoItemCount	EQ (Count(InfoItemsIn(SubFld(1.001.1))), Int(1)		M-Fld(1.001)				T

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
					)						
RT1.6 - Field: 1.001-Value	8.1.1, Table 22, 7.1, C.10.1	Field 1.001 Record header. In Traditional encoding, this field contains the record length in bytes (including all information separators). The value is unrestricted in Traditional Encoding, but must be at least 2 to accommodate the size of required fields. The XML name for the Type-1 record is <itl:PackageInformationRecord>, and its <biom:RecordCategoryCode> element shall have a value of "1".	1	1.001-Value	GTE ({Fld(1.001)}, Int(2))		M-Fld(1.001)				T
			2	1.001-Value-Dependent	EQ ({Fld(1.001)}, Count(Bytes(Rec(1))))		M-Fld(1.001)				T
			1	NIEM-1.001-Value	EQ ({XElm(1.001.biom:RecordCategoryCode)}, Str(1))		M-Fld(1.001)				X
RT1.7 - Field: 1.001-CharType	8.1 Table 22, 8	Section 8.1 and Table 22 specify the Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011, 1.012, 99.100, and 99.101. Any dates may also contain leading zeros.	1	1.001-CharType	SubSet (Chars({Fld(1.001)}), CharNum)		M-Fld(1.001)				B
			1	1.001-NoLeadingZeros	NOT (RegEx ({Fld(1.001)}, LeadingZeroNum))		M-Fld(1.001)				B
RT1.8 - Field: 1.001-CharCount	Table 22, 7.1	Table 22 specifies the character count for each field. …a minimum of 2 characters for the logical record length in Record Type-1…	1	1.001-CharCount	GTE (Count(Chars({Fld(1.001)}), Int(2))		M-Fld(1.001)				T
			1	NIEM-1.001-CharCount	EQ (Count(Chars({Fld(1.001)}), Int(1))		M-Fld(1.001)				X
RT1.9 - Field: 1.002-	Table 22, Annex B,	Table 22 specifies which fields contain subfields and information items as well as	1	1.002-VER-SubfieldCou	EQ (		M-Fld(1.002)				T

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
VER-FieldStructure	Annex G	the number of occurrences permitted. A field contains a minimum of one subfield which contains a minimum of one information item.		nt	Count(SubFldsIn(Fld(1.002))), Int(1))						
			1	1.002-VER-InfoItemCount	EQ (Count(InfoItemsIn(SubFld(1.002.1))), Int(1))		M-Fld(1.002)				T
RT1.10 - Field: 1.002-VER-Value	8.1.2	This mandatory four-character ASCII value shall be used to specify the current version number of the standard implemented by the software or system creating the transaction. The format of this field shall consist of four numeric characters. The first two characters shall specify the major version number. The last two characters shall be used to specify the minor revision number. In XML, biom:TransactionMajorVersionValue is 5 and biom:TransactionMinorVersionValue is 1	1	1.002-VER-Value	EQ ({Fld(1.002)}, Str(0501))		M-Fld(1.002)				T
			1	NIEM-1.002-VER-Value	AND (OR (EQ({XElm(1.002.biom:TransactionMajorVersionValue)}, Str(05)), EQ({XElm(1.002.biom:TransactionMajorVersionValue)}, Str(5)),) EQ(XElm(1.002.biom:TransactionMinorVersionValue), Str(01)),)		M-Fld(1.002)				X
RT1.11 - Field: 1.002-VER-CharType	8.1 Table 22, 8	Section 8.1 and Table 22 specify the Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011, 1.012, 99.100, and 99.101. Any dates may also contain leading zeros.	1	1.002-VER-CharType	SubSet (Chars({Fld(1.002)}), CharNum)		M-Fld(1.002)				B
RT1.12 - Field: 1.002-VER-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.002-VER-CharCount	EQ (Count(Chars({Fld(1.002)}), Int(4))		M-Fld(1.002)				T
			1	NIEM-1.002-VER-CharCount	AND (MO (		M-Fld(1.002)				X

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
					Count(Chars({XElm(1.002.biom:TransactionMajorVersionValue)})), Set-Int([1,2])), EQ (Count(Chars({XElm(1.002.biom:TransactionMinorVersionValue)})), Int(2)))						
RT1.13 - Field: 1.003-CNT-FieldStructure	Table 22, Annex B, Annex G	Table 22 specifies which fields contain subfields and information items as well as the number of occurrences permitted. A field contains a minimum of one subfield which contains a minimum of one information item.	1	1.003-NT-SubfieldCount	GTE (Count(SubFldsIn(Fld(1.003))), Int(2))		M-Fld(1.003)				T
			1	1.003-CNT-InfoItemCount	EQ (Count(InfoItemsIn(Fld(1.003))), Mult (Int(2), Count(SubFldsIn(Fld(1.003)))))		M-Fld(1.003)				T
			1	1.003-CNT-InfoItemStructure	InfoItemsHaveData (SubFld(1.003), Set-Int([1,2]))		M-Fld(1.003)				T
RT1.14 - Field: 1.003-FRC-Value	8.1.3	The first information item (first record category code / FRC) within this subfield shall be "1". This indicates that the first record in the transaction is a Type-1 record consisting of header information	1	1.003-FRC-Value	EQ ({InfoI(1.003.FRC)}, Int(1))		M-InfoI(1.003.FRC)				B
RT1.15 - Field: 1.003-FRC-CharType	8.1, Table 22, 8	Section 8.1 and Table 22 specify the Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011,	1	1.003-FRC-CharType	SubSet (Chars({InfoI(1.003.FRC)}), CharNum)		M-InfoI(1.003.FRC)				B
			1	1.003-FRC-	NOT		M-				B

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
		1.012, 99.100, and 99.101. Any dates may also contain leading zeros.		NoLeadingZeros	(RegEx ({InfoI(1.003.FRC)} LeadingZeroNum))		InfoI(1.003.FRC)				
RT1.16 - Field: 1.003-FRC-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.003-FRC-CharCount	EQ (Count(Chars({InfoI(1.003.FRC)}), Int(1))		M-InfoI(1.003.FRC)				B
RT1.17 - Field: 1.003-CRC-Value	8.1.3, Table 22	The second information item of this subfield (content record count / CRC) shall be the sum of the Type-2 through Type-99 records contained in this transaction. This number is also equal to the count of the remaining subfields of Field 1.003 Transaction content / CNT. The maximum value for CRC is 999.	1	1.003-CRC-Value	MO ({InfoI(1.003.CRC)}, Set-Int([1 to 999]))		M-InfoI(1.003.CRC)				B
			2	1.003-CRC-Value-Dependent-RecordCount	EQ ({InfoI(1.003.CRC)}, Count(Recs(Set-Int([2 to 99]))))		M-InfoI(1.003.CRC)				B
			2	1.003-CRC-Value-Dependent-SubfieldCount	EQ ({InfoI(1.003.CRC)}, Minus(Count(SubFldsIn(Fld(1.003))), Int(1)))		M-InfoI(1.003.CRC)				T
			2	NIEM-1.003-CRC-Value-Dependent	EQ ({XElm(1.003.biom:ContentRecordQuantity)}, Minus (Count(ElmsIn(XElm(1.003.biom:TransactionContentSummary), XElm(1.003.biom:ContentRecordSummary))), Int(1)))		M-InfoI(1.003.CRC)				X

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
RT1.18 - Field: 1.003-CRC-CharType	8.1 Table 22, 8	Section 8.1 and Table 22 specify the Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011, 1.012, 99.100, and 99.101. Any dates may also contain leading zeros.	1	1.003-CRC-CharType	SubSet (Chars({InfoI(1.003.CRC)}), CharNum)		M-InfoI(1.003.CRC)				B
			1	1.003-CRC-NoLeadingZeros	NOT (RegEx ({InfoI(1.003.CRC)}, LeadingZeroNum))		M-InfoI(1.003.CRC)				B
RT1.19 - Field: 1.003-CRC-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.003-CRC-CharCount	MO (Count(Chars({InfoI(1.003.CRC)}), Set-Int([1 to 2]))		M-InfoI(1.003.CRC)				B
RT1.20 - Field: 1.003-REC-Value	8.1.3, Table 22, Table 3	The first information item (record category code / REC), shall contain a number chosen from the “record identifier” column of Table 3.	1	1.003-REC-Value	MO ({InfoI(1.003.REC)}, Set-Int([2,4,7 to 22, 98,99]))		M-InfoI(1.003.REC)				B
RT1.21 - Field: 1.003-REC-CharType	8.1 Table 22, 8	Section 8.1 and Table 22 specify the Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011, 1.012, 99.100, and 99.101. Any dates may also contain leading zeros.	1	1.003-REC-CharType	SubSet (Chars({InfoI(1.003.REC)}), CharNum)		M-InfoI(1.003.REC)				B
			1	1.003-REC-NoLeadingZeros	NOT (RegEx ({InfoI(1.003.REC)}, LeadingZeroNum))		M-InfoI(1.003.REC)				B
RT1.22 - Field: 1.003-REC-	Table 22	Table 22 specifies the character count for each field.	1	1.003-REC-CharCount	MO (Count(Chars({InfoI(1.003.REC)}),		M-InfoI(1.003.REC)				B

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
CharCount					Set-Int([1, 2]))						
RT1.23 - Field: 1.003-IDC-Value	8.1.3, Table 22	The second information item (information designation character / IDC) shall be an integer equal to or greater than zero and less than or equal to 99. See Section 7.3.1.	1	1.003-IDC-Value	MO ({InfoI(1.003.IDC)}, Set-Int([0 to 99]))		M-InfoI(1.003.IDC)				B
RT1.24 - Field: 1.003-IDC-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011, 1.012, 99.100, and 99.101. Any dates may also contain leading zeros.	1	1.003-IDC-CharType	SubSet (Chars({InfoI(1.003.IDC)}), CharNum)		M-InfoI(1.003.IDC)				B
			1	1.003-IDC-NoLeadingZeros	NOT (RegEx ({InfoI(1.003.IDC)}, LeadingZeroNum))		M-InfoI(1.003.IDC)				B
RT1.25 - Field: 1.003-IDC-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.003-IDC-CharCount	MO (Count(Chars({InfoI(1.003.IDC)}), Set-Int([1, 2]))		M-InfoI(1.003.IDC)				B
RT1.26 - Field: 1.004-TOT-FieldStructure	Table 22, Annex B, Annex G	Table 22 specifies which fields contain subfields and information items as well as the number of occurrences permitted. A field contains a minimum of one subfield which contains a minimum of one information item.	1	1.004-TOT-SubfieldCount	EQ (Count(SubFldsIn(Fld(1.004))), Int(1))		M-Fld(1.004)				T
			1	1.004-TOT-InfoItemCount	EQ (Count(InfoItemsIn(SubFld(1.004.1))), Int(1))		M-Fld(1.004)				T
RT1.27 - Field: 1.004-TOT-Value	8.1.4, Table 22	This mandatory field shall contain an identifier, which designates the type of transaction and subsequent processing that this transaction should be given. This shall be a maximum of 16 alphabetic characters. The TOT shall be in accordance with	1	1.004-TOT-Value	ReturnResult (Result(Pass))		M-Fld(1.004)				B

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
		definitions provided by the receiving agency.) Earlier versions of this standard specifically restricted the character length of TOT to 4 characters.									
RT1.28 - Field: 1.004-TOT-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field.	1	1.004-TOT-CharType	SubSet (Chars({Fld(1.004)}), CharAlpha)		M-Fld(1.004)				B
RT1.29 - Field: 1.004-TOT-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.004-TOT-CharCount	MO (Count(Chars({Fld(1.004)}), Set-Int([1 to 16]))		M-Fld(1.004)				B
RT1.30 - Field: 1.005-DAT-FieldStructure	Table 22, Annex B, Annex G	Table 22 specifies which fields contain subfields and information items as well as the number of occurrences permitted. A field contains a minimum of one subfield which contains a minimum of one information item.	1	1.005-DAT-SubfieldCount	EQ (Count(SubFldsIn(Fld(1.005))), Int(1))		M-Fld(1.005)				T
			1	1.005-DAT-InfoItemCount	EQ (Count(InfoItemsIn(SubFld(1.005.1))), Int(1))		M-Fld(1.005)				T
RT1.31 - Field: 1.005-DAT-Value	8.1.5, Table 22, 7.7.2.3	This mandatory field shall contain the local date that the transaction was submitted. The local date is recorded as YYYYMMDD. Note that this may be a different date than the corresponding GMT, due to time zone differences.	1	1.005-DAT-Value	Complex (EQ ({Fld(1.005)}, ValidLocalDate))	t3	M-Fld(1.005)				T
			1	NIEM-1.005-DAT-Value	Complex (IfThenElse (Present(XElm(1.005.nc:Date)), EQ({XElm(1.005.nc:Date)}, NIEM-ValidLocalDate), IfThenElse (Present(XElm(1.005.nc:YearMonth)), EQ({XElm(1.005.nc:YearMonth)}, NIEM-	t3	M-Fld(1.005)				X

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
					ValidLocalYearMonth), IfThenElse (Present(XElm(1.005.nc:Year)), EQ({XElm(1.005.nc:Year)}, NIEM-ValidLocalYear), ReturnResult (Result(Fail(No valid local date element is present.)))))))						
RT1.32 - Field: 1.005-DAT-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011, 1.012, 99.100, and 99.101. Any dates may also contain leading zeros.	1	1.005-DAT-CharType	SubSet (Chars({Fld(1.005)}), CharNum)		M-Fld(1.005)				T
			1	NIEM-1.005-DAT-CharType	IfThenElse (Present(XElm(1.005.nc:Date)), SubSet (Chars({XElm(1.005.nc:Date)}) Union(CharNum, Set-Str([-]))), IfThenElse (Present(XElm(1.005.nc:YearMonth)), SubSet (Chars({XElm(1.005.nc:YearMonth)}) Union(CharNum, Set-Str([-]))), IfThenElse (Present(XElm(1.005.nc:Year)), SubSet (Chars({XElm(1.005.nc:Year)}) CharNum), ReturnResult (		M-Fld(1.005)				X

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
					Result(Fail(No valid local date element is present.))))))						
RT1.33 - Field: 1.005-DAT-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.005-DAT-CharCount	EQ (Count(Chars({Fld(1.005)}), Int(8))		M-Fld(1.005)				T
			1	NIEM-1.005-DAT-CharCount	IfThenElse (Present(XElm(1.005.nc:Date)), EQ (Count(Chars({XElm(1.005.nc:Date)})) Int(10)), IfThenElse (Present(XElm(1.005.nc:YearMonth)), EQ (Count(Chars({XElm(1.005.nc:YearMonth)})) Int(7)), IfThenElse (Present(XElm(1.005.nc:Year)), EQ (Count(Chars({XElm(1.005.nc:Year)})) Int(4)), ReturnResult (Result(Fail(No valid local date element is present.))))))		M-Fld(1.005)				X
RT1.34 - Field: 1.006-PRY-FieldStructure	Table 22, Annex B, Annex G	Table 22 specifies which fields contain subfields and information items as well as the number of occurrences permitted. A field contains a minimum of one	1	1.006-PRY-SubfieldCount	EQ (Count(SubFldsIn(Fld(1.006))), Int(1))		O-Fld(1.006)				T

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
		subfield which contains a minimum of one information item.	1	1.006-PRY-InfoItemCount	EQ (Count(InfoItemsIn(SubFld(1.006.1))), Int(1))		O-Fld(1.006)				T
RT1.35 - Field: 1.006-PRY-Value	8.1.6, Table 22	This optional field shall contain a single information character to designate the urgency with which a response is desired. The values shall range from 1 to 9, with 1 denoting the highest priority. The default value shall be defined by the agency receiving the transaction.	1	1.006-PRY-Value	{1.006} MO [1 to 9] AND MO [Integers]		O-Fld(1.006)				B
RT1.36 - Field: 1.006-PRY-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011, 1.012, 99.100, and 99.101. Any dates may also contain leading zeros.	1	1.006-PRY-CharType	SubSet (Chars({Fld(1.006)}), CharNum)		O-Fld(1.006)				B
			1	1.006-PRY-NoLeadingZeros	NOT (RegEx ({Fld(1.006)}, LeadingZeroNum))		O-Fld(1.006)				B
RT1.37 - Field: 1.006-PRY-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.006-PRY-CharCount	EQ (Count(Chars({Fld(1.006)}), Int(1))		O-Fld(1.006)				B
RT1.38 - Field: 1.007-DAI-FieldStructure	Table 22, Annex B, Annex G	Table 22 specifies which fields contain subfields and information items as well as the number of occurrences permitted. A field contains a minimum of one subfield which contains a minimum of one information item.	1	1.007-DAI-SubfieldCount	EQ (Count(SubFldsIn(Fld(1.007))), Int(1))		M-Fld(1.007)				T
			1	1.007-DAI-InfoItemCount	EQ (Count(InfoItemsIn(SubFld(1.007.1))), Int(1))		M-Fld(1.007)				T
RT1.39 - Fie	8.1.7,	This mandatory field shall contain the	1	1.007-DAI-	ReturnResult		M-Fld(1.007)				B

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
ld: 1.007-DAI-Value	Table 22	identifier of the administration or organization designated to receive the transmission. The size and data content of this field shall be user-defined and in accordance with the application profile.		Value	(Result(Pass))						
RT1.40 - Field: 1.007-DAI-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field.	1	1.007-DAI-CharType	SubSet (Chars({Fld(1.007)}), CharsAsciiPrintable)		M-Fld(1.007)				B
RT1.41 - Field: 1.007-DAI-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.007-DAI-CharCount	GTE (Count(Chars({Fld(1.007)}), Int(1))		M-Fld(1.007)				B
RT1.42 - Field: 1.008-ORI-FieldStructure	Table 22, Annex B, Annex G	Table 22 specifies which fields contain subfields and information items as well as the number of occurrences permitted. A field contains a minimum of one subfield which contains a minimum of one information item.	1	1.008-ORI-SubfieldCount	EQ (Count(SubFldsIn(Fld(1.008))), Int(1))		M-Fld(1.008)				T
			1	1.008-ORI-InfoItemCount	EQ (Count(InfoItemsIn(SubFld(1.008.1))), Int(1))		M-Fld(1.008)				T
RT1.43 - Field: 1.008-ORI-Value	8.1.8, Table 22, 5.3.1	This mandatory field shall contain the identifier of the administration or organization originating the transaction. The size and data content of this field shall be user-defined and in accordance with the application profile. The Type-1 record shall provide information describing …the originator or source of the physical record	1	1.008-ORI-Value	ReturnResult (Result(Pass))		M-Fld(1.008)				B
RT1.44 - Field: 1.008-ORI-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field.	1	1.008-ORI-CharType	SubSet (Chars({Fld(1.008)}), CharsAsciiPrintable)		M-Fld(1.008)				B
RT1.45 - Field: 1.008-	Table 22	Table 22 specifies the character count for each field.	1	1.008-ORI-CharCount	GTE (		M-Fld(1.008)				B

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
ORI-CharCount					Count(Chars({Fld(1.008)}), Int(1))						
RT1.46 - Field: 1.009-TCN-FieldStructure	Table 22, Annex B, Annex G	Table 22 specifies which fields contain subfields and information items as well as the number of occurrences permitted. A field contains a minimum of one subfield which contains a minimum of one information item.	1	1.009-TCN-SubfieldCount	EQ (Count(SubFldsIn(Fld(1.009))), Int(1))		M-Fld(1.009)				T
			1	1.009-TCN-InfoItemCount	EQ (Count(InfoItemsIn(SubFld(1.009.1))), Int(1))		M-Fld(1.009)				T
RT1.47 - Field: 1.009-TCN-Value	8.1.9, Table 22	This mandatory field shall contain the transaction control number as assigned by the originating agency. A unique (for the originating agency) alphanumeric control number shall be assigned to each transaction. For any transaction that requires a response, the respondent shall refer to this number in communicating with the originating agency.	1	1.009-TCN-Value	ReturnResult (Result(Pass))		M-Fld(1.009)				B
RT1.48 - Field: 1.009-TCN-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field.	1	1.009-TCN-CharType	SubSet (Chars({Fld(1.009)}), CharsAsciiPrintable)		M-Fld(1.009)				B
RT1.49 - Field: 1.009-TCN-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.009-TCN-CharCount	GTE (Count(Chars({Fld(1.009)}), Int(1))		M-Fld(1.009)				B
RT1.50 - Field: 1.010-TCR-FieldStructure	Table 22, Annex B, Annex G	Table 22 specifies which fields contain subfields and information items as well as the number of occurrences permitted. A field contains a minimum of one subfield which contains a minimum of one information item.	1	1.010-TCR-SubfieldCount	EQ (Count(SubFldsIn(Fld(1.010))), Int(1))		O-Fld(1.010)				T
			1	1.010-TCR-InfoItemCount	EQ (Count(InfoItemsIn(SubFld(1.010.1))), Int(1))		O-Fld(1.010)				T

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
RT1.51 - Field: 1.010-TCR-Value	8.1.10, Table 22	This optional field shall be used for responses that refer to the TCN of a previous transaction involving an inquiry or other action that required a response.	1	1.010-TCR-Value	ReturnResult (Result(Pass))		O-Fld(1.010)				B
RT1.52 - Field: 1.010-TCR-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field.	1	1.010-TCR-CharType	SubSet (Chars({Fld(1.010)}), CharsAsciiPrintable)		O-Fld(1.010)				B
RT1.53 - Field: 1.010-TCR-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.010-TCR-CharCount	GTE (Count(Chars({Fld(1.010)}), Int(1))		O-Fld(1.010)				B
RT1.54 - Field: 1.011-NSR-FieldStructure	Table 22, Annex B, Annex G	Table 22 specifies which fields contain subfields and information items as well as the number of occurrences permitted. A field contains a minimum of one subfield which contains a minimum of one information item.	1	1.011-NSR-SubfieldCount	EQ (Count(SubFldsIn(Fld(1.011))), Int(1))		M-Fld(1.011)				T
			1	1.011-NSR-InfoItemCount	EQ (Count(InfoItemsIn(SubFld(1.011.1))), Int(1))		M-Fld(1.011)				T
RT1.55 - Field: 1.011-NSR-Value	8.1.11, Table 22, 7.7.6, 7.7.6.1, 7.7.6.2.1, Table 14	This mandatory field shall be set to "00.00" if there are no Type-4 records in the transaction. When there are Type-4 records present, this field is used to specify the native scanning resolution of the friction ridge image capture device. This field shall specify the resolution in pixels per millimeter. The resolution shall be expressed as two numeric characters followed by a decimal point and two more numeric characters. Images with scanning resolution greater than or equal to the 1000 ppi class should not be transmitted using Record Type-4 unless being transmitted at 500 ppi class to a system incapable of receiving Type-14	1	1.011-NSR-Value	RegEx ({Fld(1.011)}, Str(^[0-9]{2}\.[0-9]{2}$))		M-Fld(1.011)				T
			1	NIEM-1.011-NSR-Value	RegEx ({Fld(1.011)}, Str(^[0-9]{1,2}\.[0-9]{2}$))		M-Fld(1.011)				X
			2	1.011-NSR-Value-Dependent	IfThenElseResult (Not(Present(Rec(4))), EQ({Fld(1.011)}, Num(00.00)), IfThenElseResult (GTE(NV({Fld(1.011)}), Num(38.58)), ReturnResult	t11	M-Fld(1.011)	REMOVE NIEM ASSERTION BELOW			B

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
		records at 1000 ppi class or greater. NSR contains five characters specifying the native scanning resolution in pixels per millimeter. It is expressed as two numeric characters followed by a decimal point and two more numeric characters (e.g. 19.69). Exemplar images shall have a minimum scanning resolution of the 500 ppi class. In this version, NSR and NTR only apply to Record Type-4: Grayscale fingerprint image… Table 14 defines resolution tolerance for fingerprint types. 2% is used as the default; see test note t-11 for details. Note: the minimum value with tolerance was 19.30 in 2011. This is changed to 19.29 in 2013 due to the rounding method mentioned in 7.7.8.4.			(Result(Warning(Images with scanning resolution equal to or greater than the 1000ppi class should not be transmitted using Record Type-4 unless they are scaled-down to produce a transmitting resolution of class 500ppi.))), GTE(NV({Fld(1.011)}), Num(19.29)),))						
RT1.56 - Field: 1.011-NSR-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011, 1.012, 99.100, and 99.101. Any dates may also contain leading zeros.	1	1.011-NSR-CharType	SubSet (Chars({Fld(1.011)}), Union(CharNum, Set-Str([.])))		M-Fld(1.011)				B
RT1.57 - Field: 1.011-NSR-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.011-NSR-CharCount	EQ (Count(Chars({Fld(1.011)})), Int(5))		M-Fld(1.011)				T
			1	NIEM-1.011-NSR-CharCount	MO (Count(Chars({Fld(1.011)})), Set-Int([4,5])		M-Fld(1.011)				X

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
					)						
RT1.58 - Field: 1.012-NTR-FieldStructure	Table 22, Annex B, Annex G	Table 22 specifies which fields contain subfields and information items as well as the number of occurrences permitted. A field contains a minimum of one subfield which contains a minimum of one information item.	1	1.012-NTR-SubfieldCount	EQ (Count(SubFldsIn(Fld(1.012))), Int(1))		M-Fld(1.012)				T
			1	1.012-NTR-InfoItemCount	EQ (Count(InfoItemsIn(SubFld(1.012.1))), Int(1))		M-Fld(1.012)				T
RT1.59 - Field: 1.012-NTR-Value	8.1.12, Table 22, 7.7.6, 7.7.6.3.1	This mandatory field shall be set to "00.00" if there are no Type-4 records in the transaction. When there are Type-4 records present, this field specifies the nominal resolution for the image(s) being exchanged. This field shall specify the resolution in pixels per millimeter. The resolution shall be within the range 19.30 ppmm (490 ppi) to 20.08 ppmm (510 ppi). All record types containing images are variable resolution except for Type-4, which has a fixed resolution. Record Type-4 shall not be used for anything but the 500 ppi class. In this version, NSR and NTR only apply to Record Type-4: Grayscale fingerprint image… …the transmitting resolution shall not be greater than the scanning resolution	1	1.012-NTR-Value	RegEx ({Fld(1.012)}, Str(^[0-9]{2}\.[0-9]{2}$))		M-Fld(1.012)				T
			1	NIEM-1.012-NTR-Value	RegEx ({Fld(1.012)}, Str(^[0-9]{1,2}\.[0-9]{2}$))		M-Fld(1.012)				X
			2	1.012-Value-Dependent	IfThenElse (Present(Rec(4)), InRange (NV({Fld(1.012)}), Num(19.29), Num(20.08)), EQ({Fld(1.012)}, Num(00.00)))	t11	M-Fld(1.012)	REMOVE NIEM ASSERT BELOW			B
			2	1.012-NTR-Value-Dependent-LTE-1.011	LTE ({Fld(1.012)}, {Fld(1.011)})		M-Fld(1.012)				B
RT1.60 - Field: 1.012-NTR-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011, 1.012, 99.100, and 99.101. Any dates may	1	1.012-NTR-CharType	SubSet (Chars({Fld(1.012)}), Union(CharNum, Set-Str([.])))		M-Fld(1.012)				B

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
		also contain leading zeros.									
RT1.61 - Field: 1.012-NTR-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.012-NTR-CharCount	EQ (Count(Chars({Fld(1.012)})), Int(5))		M-Fld(1.012)				T
			1	NIEM-1.012-NTR-CharCount	MO (Count(Chars({Fld(1.012)})), Set-Int([4,5]))		M-Fld(1.012)				X
RT1.62 - Field: 1.013-DOM-FieldStructure	Table 22, Annex B, Annex G	Table 22 specifies which fields contain subfields and information items as well as the number of occurrences permitted. A field contains a minimum of one subfield which contains a minimum of one information item.	1	1.013-DOM-SubfieldCount	EQ (Count(SubFldsIn(Fld(1.013))), Int(1))		O-Fld(1.013)				T
			1	1.013-DOM-InfoItemCount	EQ (Count(InfoItemsIn(SubFld(1.013.1))), Int(2))		O-Fld(1.013)				T
			1	1.013-DOM-InfoItemStructure	InfoItemsHaveData (SubFld(1.013.1), Set-Int([1]))		O-Fld(1.013)				T
RT1.63 - Field: 1.013-DNM-Value	8.1.13, Table 22	The mandatory first information item (domain name / DNM) will uniquely identify the agency, entity, or implementation used for formatting the fields in the Type-2 record. The default value for the field shall be the North American Domain implementation (NORAM).	1	1.013-DNM-Value	ReturnResult (Result(Pass))		M↑-InfoI(1.013.DNM)				B
RT1.64 - Field: 1.013-DNM-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field.	1	1.013-DNM-CharType	SubSet (Chars({InfoI(1.013.DNM)}), CharsAsciiPrintable)		M↑-InfoI(1.013.DNM)				B
RT1.65 - Fie	Table 22	Table 22 specifies the character count for	1	1.013-DNM-	GTE		M↑-				B

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
ld: 1.013-DNM-CharCount		each field.		CharCount	(Count(Chars({InfoI(1.013.DNM)}), Int(1))		InfoI(1.013.DNM)				
RT1.66 - Field: 1.013-DVN-Value	8.1.13, Table 22	An optional second information item (domain version number / DVN) shall contain the unique version of the particular implementation, such as 7.02.	1	1.013-DVN-Value	ReturnResult (Result(Pass))		O↑-InfoI(1.013.DVN)				B
RT1.67 - Field: 1.013-DVN-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field.	1	1.013-DVN-CharType	SubSet (Chars({InfoI(1.013.DVN)}), CharsAsciiPrintable)		O↑-InfoI(1.013.DVN)				B
RT1.68 - Field: 1.013-DVN-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.013-DVN-CharCount	GTE (Count(Chars({InfoI(1.013.DVN)}), Int(1))		O↑-InfoI(1.013.DVN)				B
RT1.69 - Field: 1.014-GMT-FieldStructure	Table 22, Annex B, Annex G	Table 22 specifies which fields contain subfields and information items as well as the number of occurrences permitted. A field contains a minimum of one subfield which contains a minimum of one information item.	1	1.014-GMT-SubfieldCount	EQ (Count(SubFldsIn(Fld(1.014))), Int(1))		O-Fld(1.014)				T
			1	1.014-GMT-InfoItemCount	EQ (Count(InfoItemsIn(SubFld(1.014.1))), Int(1))		O-Fld(1.014)				T
RT1.70 - Field: 1.014-GMT-Value	8.1.14, Table 22	This optional field provides a mechanism for expressing the date and time in terms of universal Greenwich Mean Time (GMT) units.	1	1.014-GMT-Value	Complex (EQ ({Fld(1.014)}, ValidUTC/GMT))	t3	O-Fld(1.014)				T
			1	NIEM-1.014-GMT-Value	Complex (EQ {XElm(1.014.nc:DateTime)}, NIEM-ValidUTC/GMT))	t3	O-Fld(1.014)				X
RT1.71 - Fie	8.1	Section 8.1 and Table 22 specify the	1	1.014-GMT-	SubSet		O-Fld(1.014)				B

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
ld: 1.014-GMT-CharType	Table 22,	Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011, 1.012, 99.100, and 99.101. Any dates may also contain leading zeros.		CharType	(Chars({Fld(1.014)}), Union(CharNum, Set-Str([Z])))						
			1	NIEM-1.014-GMT-CharType	SubSet (Chars({Fld(1.014)}), Union(CharNum, Set-Str([-,:,T,Z])))		O-Fld(1.014)				B
RT1.72 - Field: 1.014-GMT-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.014-GMT-CharCount	EQ (Count(Chars({Fld(1.014)}), Int(15))		O-Fld(1.014)				T
			1	NIEM-1.014-GMT-CharCount	EQ (Count(Chars({Fld(1.014)}), Int(20))		O-Fld(1.014)				X
RT1.73 - Field: 1.015-DCS-FieldStructure	Table 22, Annex B, Annex G	Table 22 specifies which fields contain subfields and information items as well as the number of occurrences permitted. A field contains a minimum of one subfield which contains a minimum of one information item.	1	1.015-DCS-SubfieldCount	EQ (Count(SubFldsIn(Fld(1.015))), Int(1))		O-Fld(1.015)				T
			1	1.015-DCS-InfoItemCount	EQ (Count(InfoItemsIn(SubFld(1.015.1))), Int(2,3))		O-Fld(1.015)				T
			1	1.015-DCS-InfoItemStructure	InfoItemsHaveData (SubFld(1.015.1), Set-Int([1,2]))		O-Fld(1.015)				T
RT1.74 - Field: 1.015-CSI-Value	8.1.15, Table 22, Table 4, 5.4, 5.6	The first information item (character encoding index / CSI) is the index number that references an associated character encoding. See the "Character encoding index" column of Table 4 for the valid values for this information item.	1	1.015-CSI-Value	IfThenElseResult (MO(NV{InfoI(1.015.CSI)}, Set-Int([0, 2 to 4, 128 to 999]), Result(Pass), IfThenElseResult (		M↑-InfoI(1.015.CSI)				B

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
		'Legacy' indicates that if there is existing data using this record type, field, information item or value it may still be transmitted in a transaction conformant to this version of the standard. In this version 'legacy' applies to Fields 9.005 through 9.012, Field 10.022 and to the value '1' in Table 4 Character encoding. Note that the value "1" does not appear in the table. It is a legacy value.			EQ({InfoI(1.015.CSI)}, Str(1)), Result(Warning('1' is a Legacy value.)), Result(Fail)))						
RT1.75 - Field: 1.015-CSI-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011, 1.012, 99.100, and 99.101. Any dates may also contain leading zeros.	1	1.015-CSI-CharType	SubSet (Chars({InfoI(1.015.CSI)}), CharNum)		M↑-InfoI(1.015.CSI)				B
			1	1.015-CSI-NoLeadingZeros	NOT (RegEx ({InfoI(1.015.CSI)}, LeadingZeroNum))		M↑-InfoI(1.015.CSI)				B
RT1.76 - Field: 1.015-CSI-CharType	Table 22	Table 22 specifies the character count for each field.	1	1.015-CSI-CharCount	MO (Count(Chars({InfoI(1.015.CSI)}), Set-Int([1,2,3]))		M↑-InfoI(1.015.CSI)				B
RT1.77 - Field: 1.015-CSN-Value	8.1.15, Table 22, Table 4	The second information item (character encoding name / CSN) shall be the "Character encoding name" associated with that index number, taken from Table 4.	1	1.015-CSN-Value	ReturnResult (Result(Pass))		M↑-InfoI(1.015.CSN)				B
			2	1.015-CSN-Value-Dependent	IfThenElseResult (EQ({InfoI(1.015.CSI)}, Int(0)), EQ({InfoI(1.015.CSN)}, Str(ASCII)), IfThenElseResult (EQ({InfoI(1.015.CSI)}, Int(1)),		M↑-InfoI(1.015.CSN)				X

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
					EQ({InfoI(1.015.CSN)}, Str(8-bit ASCII)), IfThenElseResult (EQ({InfoI(1.015.CSI)}, Int(2)), EQ({InfoI(1.015.CSN)}, Str(8 UTF-16)), IfThenElseResult (EQ({InfoI(1.015.CSI)}, Int(3)), EQ({InfoI(1.015.CSN)}, Str(8 UTF-8)), IfThenElseResult (EQ({InfoI(1.015.CSI)}, Int(4)), EQ({InfoI(1.015.CSN)}, Str(8 UTF-32)), ReturnResult (Result(Pass)))))))						
RT1.78 - Field: 1.015-CSN-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011, 1.012, 99.100, and 99.101. Any dates may also contain leading zeros.	1	1.015-CSN-CharType	SubSet (Chars({InfoI(1.015.CSN)}), CharAsciiPrintable)		M↑-InfoI(1.015.CSN)				B
RT1.79 - Field: 1.015-CSN-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.015-CSN-CharCount	MO (Count(Chars({InfoI(1.015.CSN)}), Set-Int([1 to 16]))		M↑-InfoI(1.015.CSN)				B
RT1.80 - Field: 1.015-CSV-Value	8.1.15, Table 22, Table 4	The optional third information item (character encoding version / CSV) is the specific version of the character encoding used. In the case of the use of UTF-8, the third optional information item may be used to hold the specific version used, so that the display terminal can be switched to the correct font family.	1	1.015-CSV-Value	ReturnResult (Result(Pass))		O↑-InfoI(1.015.CSV)				B

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
RT1.81 - Field: 1.015-CSV-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011, 1.012, 99.100, and 99.101. Any dates may also contain leading zeros.	1	1.015-CSV-CharType	SubSet (Chars({InfoI(1.015.CSV)}), CharAsciiPrintable)		O↑-InfoI(1.015.CSV)				B
RT1.82 - Field: 1.015-CSV-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.015-CSV-CharCount	MO (Count(Chars({InfoI(1.015.CSV)}), Set-Int([1 to 16]))		M↑-InfoI(1.015.CSV)				B
RT1.83 - Field: 1.016-APS-FieldStructure	Table 22, Annex B, Annex G	Table 22 specifies which fields contain subfields and information items as well as the number of occurrences permitted. A field contains a minimum of one subfield which contains a minimum of one information item.	1	1.016-APS-SubfieldCount	MO (Count(SubFldsIn(Fld(1.016))), Set-Int(]1 to 99]))		O-Fld(1.016)				T
			1	1.016-APS-InfoItemCount	EQ (Count(InfoItemsIn(SubFld(1.016))), Int(3))		O-Fld(1.016)				T
			1	1.016-APS-InfoItemStructure	InfoItemsHaveData (SubFld(1.016), Set-Int([1,2,3]))		O-Fld(1.016)				T
RT1.84 - Field: 1.016-APO-Value	8.1.16, Table 22	The first information item (application profile organization / APO) will uniquely identify the agency or entity responsible for the specification.	1	1.016-APO-Value	ReturnResult (Result(Pass))		M↑-InfoI(1.016.APO)				B
RT1.85 - Field: 1.016-APO-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011, 1.012, 99.100, and 99.101. Any dates may	1	1.016-APO-CharType	SubSet (Chars({InfoI(1.016.APO)}), CharAsciiPrintable)		M↑-InfoI(1.016.APO)				B

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
		also contain leading zeros.									
RT1.86 - Field: 1.016-APO-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.016-APO - CharCount	GTE (Count(Chars({InfoI(1.016.APO}), Int(1))		M↑-InfoI(1.016.APO)				B
RT1.87 - Field: 1.016-APN-Value	8.1.16, Table 22	The second information item (application profile name / APN) shall contain the name of the specification.	1	1.016-APN-Value	ReturnResult (Result(Pass))		M↑-InfoI(1.016.APN)				B
RT1.88 - Field: 1.016-APN-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011, 1.012, 99.100, and 99.101. Any dates may also contain leading zeros.	1	1.016-APN-CharType	SubSet (Chars({InfoI(1.016.APN)}), CharAsciiPrintable)		M↑-InfoI(1.016.APN)				B
RT1.89 - Field: 1.016-APN-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.016-APN - CharCount	GTE (Count(Chars({InfoI(1.016.APN}), Int(1))		M↑-InfoI(1.016.APN)				B
RT1.90 - Field: 1.016-APV-Value	8.1.16, Table 22	The third information item (application profile version number / APV) shall contain the specific version of the specification.	1	1.016-APV-Value	ReturnResult (Result(Pass))		M↑-InfoI(1.016.APV)				B
RT1.91 - Field: 1.016-APV-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011, 1.012, 99.100, and 99.101. Any dates may also contain leading zeros.	1	1.016-APV-CharType	SubSet (Chars({InfoI(1.016.APV)}), CharAsciiPrintable)		M↑-InfoI(1.016.APV)				B
RT1.92 - Field: 1.016-APV-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.016-APV - CharCount	GTE (Count(Chars({InfoI(1.016.APV}), Int(1))		M↑-InfoI(1.016.APV)				B
RT1.93 - Field: 1.016-	8.1.16	If multiple Application Profile Specifications are included in this field, the	3	1.016-APS Compliance	ReturnResult (		O-Fld(1.016)				B

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
APS-Compliance		specifications must be compatible with each other: this transaction must be in compliance with all of the cited specifications. See Section 6.			Result (Warning(Untested Level 3-Application Profiles external references are outside of the scope of conformance testing to the base standard.)))						
RT1.94 - Field: 1.017-ANM-FieldStructure	Table 22, Annex B, Annex G	Table 22 specifies which fields contain subfields and information items as well as the number of occurrences permitted. A field contains a minimum of one subfield which contains a minimum of one information item.	1	1.017-ANM-SubfieldCount	EQ (Count(SubFldsIn(Fld(1.017))), Int(1))		O-Fld(1.017)				T
			1	1.017-ANM-InfoItemCount	LTE (Count(InfoItemsIn(SubFld(1.017.1))), Int(2))		O-Fld(1.017)				T
			1	1.017-ANM-InfoItemCount	ReturnResult (Result(Pass(All Information Items are optional for this field.)))		O-Fld(1.017)				T
RT1.95 - Field:1.017-DAN-Value	8.1.17	Both information items are alphanumeric and can have any special characters in the names.	1	1.017-DAN-Value	ReturnResult (Result(Pass))		O↑-InfoI(1.017.DAN)				B
RT1.96 - Field:1.017-DAN-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011, 1.012, 99.100, and 99.101. Any dates may also contain leading zeros.	1	1.017-DAN-CharType	SubSet (Chars({InfoI(1.017.DAN)}), CharAsciiPrintable)		O↑-InfoI(1.017.DAN)				B
RT1.97 - Field:1.017-DAN-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.017-DAN-CharCount	GTE (Count(Chars({InfoI(1.017.DAN}), Int(1))		O↑-InfoI(1.017.DAN)				B
RT1.98 - Field:1.017-OAN-Value	8.1.17	Both information items are alphanumeric and can have any special characters in the names.	1	1.017-OAN-Value	ReturnResult (Result(Pass))		O↑-InfoI(1.017.OAN)				B
RT1.99 - Fie	8.1	Section 8.1 and Table 22 specify the	1	1.017-OAN-	SubSet		O↑-				B

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
ld:1.017-OAN-CharType	Table 22,	Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011, 1.012, 99.100, and 99.101. Any dates may also contain leading zeros.		CharType	(Chars({InfoI(1.017.OAN)}), CharAsciiPrintable)		InfoI(1.017.OAN)				
RT1.100 - Field:1.017-OAN-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.017-OAN-CharCount	GTE (Count(Chars({InfoI(1.017.OAN}), Int(1))		O↑-InfoI(1.017.OAN)				B
RT1.101 - Field: 1.018-GNS-FieldStructure	Table 22, Annex B, Annex G	Table 22 specifies which fields contain subfields and information items as well as the number of occurrences permitted. A field contains a minimum of one subfield which contains a minimum of one information item.	1	1.018-GNS-SubfieldCount	EQ (Count(SubFldsIn(Fld(1.018))), Int(1))		O-Fld(1.018)				T
			1	1.018-GNS-InfoItemCount	EQ (Count(InfoItemsIn(SubFld(1.018.1))), Int(1))		O-Fld(1.018)				T
RT1.102 - Field: 1.018-GNS-Value	8.1.18	This optional field is used if the transaction uses GENC in lieu of ISO 3166-1 as a code set for country code specifications. ISO 3166-1 is the default country code set used for the transaction when this field is not contained in Record Type-1. The values for this field are: ISO, GENC	1	1.018-GNS-Value	MO ({Fld(1.018)}, Set-Str([ISO, GENC]))		O-Fld(1.018)				B
RT1.103 - Field: 1.018-GNS-CharType	8.1 Table 22,	Section 8.1 and Table 22 specify the Character Type for each field. Numeric values shall not contain leading zeros unless indicated by the standard text. Leading zeros are allowed for 1.002, 1.011, 1.012, 99.100, and 99.101. Any dates may also contain leading zeros.	1	1.018-GNS-CharType	SubSet (Chars({Fld(1.018.GNS)}), CharAlpha)		O-Fld(1.018)				B
RT1.104 - Field: 1.018-GNS-CharCount	Table 22	Table 22 specifies the character count for each field.	1	1.018-GNS-CharType	MO (Count(Chars({Fld(1.018})), Set-Int([3,4])		O-Fld(1.018)				B

Req. # - ID	Ref. in Base Std.	Requirement Summary	Level	Assertion ID	Test Assertion	Notes	Imp. Required	Imp. Support	Supported Range	Test Result	Enc.
Record Type-1: Transaction information record											
					)						

Annex D: Test Notes and Test Exceptions

This Annex defines test notes and test exceptions that apply to requirements and assertions documented in Annex C as well as test notes and exceptions for those requirements which may be released in separate publications. As these test notes and requirements may need to be updated as additional requirements are documented, the test notes and test exceptions are included as an external reference. The test notes and exceptions will be made available at: http://www.nist.gov/itl/csd/biometrics/biocta_download.cfm.

Acknowledgements

This publication was the result of work was sponsored, in part, by the Department of Homeland Security/Office of Biometric Identity Management (OBIM). Christofer J. McGinnis, from IDTP, a NIST/ITL grantee, developed most of the test assertions documented in this publication.