



Intelligence Community Technical Specification

XML Data Encoding Specification for Trusted Data Format - Base

Version 2021-NOV

December 1, 2022

Distribution Notice:

This document has been approved for Public Release and is available for use without restriction.

Table of Contents

Chapter 1 - Introduction	1
1.1 - Purpose	1
1.2 - Scope	1
1.3 - Enterprise Need	1
1.4 - Conventions	2
1.4.1 - XML Namespaces	2
1.5 - Dependencies	2
1.5.1 - Specification Dependencies	2
1.5.2 - Inverse Dependencies	5
Chapter 2 - Development Guidance	6
2.1 - TDF Structure	6
2.1.1 - Version Declarations	9
2.2 - Assertions	9
2.2.1 - Assertion Scopes	9
2.2.1.1 - Assertion Scopes Within TDO	10
2.2.1.2 - Assertion Scopes Within TDC	10
2.2.1.3 - HandlingAssertion scopes within TDO	11
2.2.1.4 - HandlingAssertion scopes within TDC	12
2.2.2 - Mission-Specific Metadata Assertions	12
2.2.3 - Assertions and Data State	12
2.3 - Binding and BindingInfo	12
2.4 - Normalization Method	17
2.5 - Encryption and EncryptionInfo	17
2.6 - Linked or Embedded Data Objects	18
2.7 - MIME type	18
2.8 - Payload Chunking	18
2.9 - BASE-TDF Schematron Rules	18
Chapter 3 - Constraints	19
3.1 - Data Validation Constraint Rules	19
3.1.1 - Purpose	19
3.1.2 - Value Enumeration Constraints	19
3.1.3 - Additional Constraints	19
3.1.3.1 - DES Constraints	19
3.1.4 - Constraint Rules	19
3.2 - Data Rendering Constraint Rules	20
3.2.1 - Purpose	20
3.2.2 - Rendering Constraint Rules	20
Chapter 4 - Conformance Validation	21
4.1 - Definitions	21
4.2 - Why a verbose validation strategy is required	21
4.3 - Required Order of HandlingAssertions	22
4.4 - TDO Validation Steps	23
4.4.1 - Step 1 - TDO aware and cross Assertion constraints	23
4.4.2 - Step 2 - Extension point constraints	23
4.5 - TDC Validation Steps	24
4.5.1 - Step 1 - TDC aware and cross Assertion constraints	24

4.5.2 - Step 2 - Extension point constraints	24
4.5.3 - Step 3 - Recursive Validation	24
Chapter 5 - Future Features	25
5.1 - Explicit Scope	25
5.2 - BoundValueList	25
Appendix A - Feature Summary	26
A.1 - BASE-TDF Feature Summary	26
Appendix B - Change History	27
B.1 - V2021-NOV Release Summary	27
B.2 - V2021-JAN Initial Release Summary	27
Appendix C - List of Abbreviations	29
Appendix D - Bibliography	30
Appendix E - Points of Contact	32
Appendix F - IC CIO Approval Memo	33

List of Figures

Figure 1 - Related Specifications	4
Figure 2 - Inverse Dependency Specifications	5
Figure 3 - Simple TDO	7
Figure 4 - TDO with Encryption	7
Figure 5 - TDF Structure	8
Figure 6 - Trusted Data Collection (TDC)	9
Figure 7 - TDF Extension Points	22

List of Tables

Table 1 - XML Namepaces	2
Table 2 - Direct Dependencies	3
Table 3 - TDO Binding Contents	13
Table 4 - TDC Binding Contents	15
Table 5 - Sample URLs for XML Canonicalization Normalization Methods	17
Table 6 - Constraint Rules	20
Table 7 - Feature Summary Legend	26
Table 8 - BASE-TDF Feature comparison	26
Table 9 - DES Version Identifier History	27
Table 10 - Data Encoding Specification V2021-NOV Initial Release Summary	27
Table 11 - Data Encoding Specification V2021-JAN Initial Release Summary	28

Chapter 1 - Introduction

1.1 - Purpose

This *XML Data Encoding Specification for Trusted Data Format - Base* (BASE-TDF.XML) defines detailed implementation guidance for using Extensible Markup Language (XML) to encode BASE-TDF data. This Data Encoding Specification (DES) defines the XML elements and attributes, associated structures and relationships, mandatory and cardinality requirements, and permissible values for representing BASE-TDF data concepts using XML.

1.2 - Scope

The *Intelligence Community Technical Specification Framework* (IC-SF.XML^[2]) defines the basic conceptual structure and outlines the core philosophy of Intelligence Community (IC) technical specifications. For convenience, a copy of this framework is included in every package.

This specification is applicable to the IC and information produced by, stored, or shared within the IC. This DES may have relevance outside the scope of intelligence; however, prior to applying outside of this defined scope, the DES should be closely scrutinized and differences separately documented and assessed for applicability.

1.3 - Enterprise Need

Information sharing within the national intelligence enterprise will increasingly rely on information assurance metadata (including enterprise data headers) to allow interagency access control, automated exchanges, and appropriate protection of shared intelligence. A structured, verifiable representation of security metadata bound to the intelligence data is required in order for the enterprise to become inherently "smarter" about the information flowing in and around it. Such a representation, when implemented with other data formats, improved user interfaces, and data processing utilities, can provide part of a larger, robust information assurance infrastructure capable of automating some of the management and exchange decisions today being performed by human beings.

Both enterprise needs and requirements for this specification can be found in the following policies and implementation guidance:

- 200 Series:
 - Intelligence Community Directive (ICD) 208, *Write for Maximum Utility*^[4]
 - ICD 209, *Tearline Production and Dissemination*^[5]
 - Intelligence Community Policy Memorandum (ICPM) 2007-200-2, *Preparing Intelligence to Meet the Intelligence Community's Responsibility to Provide*^[8]
- 500 Series:
 - ICD 500, *Director Of National Intelligence Chief Information Officer*^[6]
 - ICD 501, *Discovery and Dissemination or Retrieval of Information within the IC*^[7]
 - Intelligence Community Standard (ICS) 500-20, *IC Enterprise Standards Compliance*^[9]
 - ICS 500-21, *Tagging of Intelligence and Intelligence-Related Information*^[10]

1.4 - Conventions

Certain technical and presentation conventions are used in the creation of the IC technical specifications to ensure readability and understanding. For details, please see the “Specification Conventions” chapter in the IC-SF.XML^[2].

1.4.1 - XML Namespaces

Namespaces referenced in this document and the prefixes used to represent them are listed in the following table. The namespace prefix of any XML Qualified Name used in any example in this document should be interpreted using the information below.

Table 1 - XML Namepaces

Prefix	URI
tdf	urn:us:gov:ic:tdf

1.5 - Dependencies

Specifications often rely on other specifications, components or artifacts, either directly or indirectly. For specific definitions of dependency terminology used throughout this section, please see the “Dependency Definitions” chapter in the IC-SF.XML^[2].

1.5.1 - Specification Dependencies

This technical specification directly depends on the technical specifications, documentation, and implementations listed in [Table 2](#). The dependencies listed below are directly referenced in this specification (e.g., Schema, Schematron), and are normative or informative as indicated.

The subsequent figure, [Figure 1](#), is an informative graphical representation of all of the Intelligence Community Chief Information Officer (IC CIO) specifications related to this specification. The graphic depicts dependencies. However, the representations may not match an exact schema import tree or dependency diagram that an analysis of the Schema, Schematron or other documents would yield. For example, the graphic only shows a given specification once even though it may actually be imported by many specifications or be a direct dependency. All IC CIO specifications listed in [Table 2](#) will be shown in [Figure 1](#); however not all IC CIO specifications listed in [Figure 1](#) may appear in [Table 2](#). [Figure 1](#) is to aid users in gaining a general understanding of all dependencies whether direct or transitive.

Table 2 - Direct Dependencies

Name	Dependency Description
<p><i>Intelligence Community Specification Framework</i> (IC-SF.XML.V2021-NOV+^[2])</p>	<p>This specification depends on a specific version of IC-SF.XML^[2]; versions later than version 2021-NOV MAY be used, however, the newest version of IC-SF.XML SHOULD be used as IC-SF.XML is expected to always replace its preceding version. The minimum version was based on a technical dependency; the creation of schema fragments.</p>
<p>Schematron^[12]</p>	<p>Schematron — International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 19757-3:2006 — is a rule-based document schema definition language. In this specification Schematron is a formal language used to express normative business rules, so this reference is normative.</p> <p>The Schematron rules are normative in the sense that they convey criteria that a document MUST adhere to, exactly as English may be used to convey normative criteria. It is not necessary for implementers to use the specific Schematron encoding in this specification. Implementers MAY use any encodings, tools, or languages desired to implement validation schemes for conformance to this specification.</p> <p>Note: The Schematron rules in this specification use Transformations (XSLT) 2.0^[13] query binding.</p>
<p>XSLT 2.0^[13] implementation of Schematron^[12] by Rick Jelliffe (2010-04-14)</p> <p>Note: The only available identifying descriptors for this implementation are the implementer's name and date of release. This implementation may be found at the following Uniform Resource Locator (URL): http://code.google.com/p/schematron/.</p>	<p>The International Organization for Standardization does not create nor endorse reference implementations of its standards. For the purposes of this specification the <i>behavior</i> of the implementation created by Mr. Jelliffe is normative.</p> <p>Implementers MAY use any encodings, tools, or languages desired to implement validation schemes for conformance to this specification. To conform to this specification, a validator MUST find a document valid <i>if and only if</i> the Schematron implementation by Mr. Jelliffe would find the document valid according to the Schematron rules in this specification.</p>

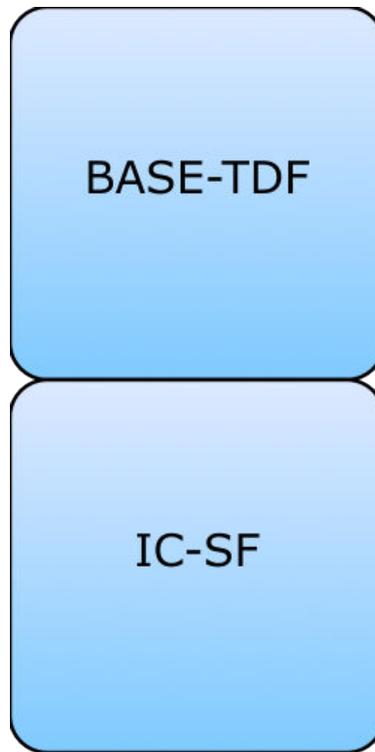


Figure 1 : Related Specifications

1.5.2 - Inverse Dependencies

Generally, it is only necessary to think of the *dependencies* in the dependency tree. However, with the specification versions being decoupled, it is also important to consider the *inverse dependencies*, for compatibility with newer versions of a given specification. The changes introduced to a given specification can sometimes make it incompatible with current versions of its inverse dependencies (specifications that uses the given specification).

Since this specification is one such specification that is used by other specifications released by the IC CIO, the [Figure 2](#) has been included to assist readers in understanding all of the inverse dependency relationships and how changes in this given specification may impact others specifications. This diagram is representative of direct and transitive inverse dependencies at the time of the release of this specification, but are subject to change over time and is presented in a list format that is different than [Figure 1](#).

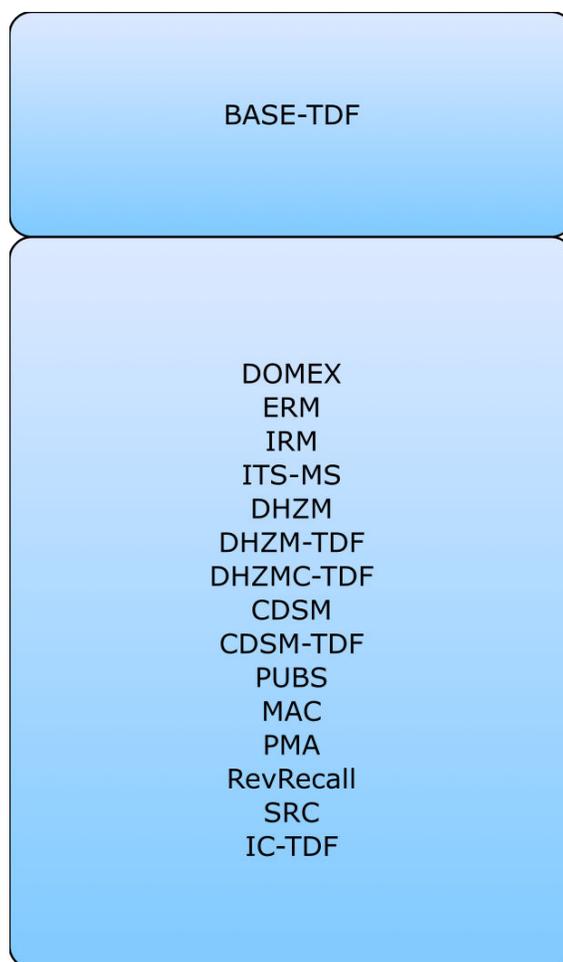


Figure 2 : Inverse Dependency Specifications

Chapter 2 - Development Guidance

For information on the structure and content of the specifications, please see the "Specification Overview" chapter in the IC-SF.XML^[2] framework document. This chapter is intended to expand upon the common information that the framework specifies providing specific development guidance that is specific to the implementation of this specification.

BASE-TDF.XML is the top-level specification from which all other Trusted Data Format (TDF) specifications are generated. TDF specifications, such as *XML Data Encoding Specification for Trusted Data Format* (IC-TDF.XML^[3]), inherit only those capabilities that are needed to fulfill its requirements. Child TDF instances will always validate against the parent BASE-TDF.XML schema, however, BASE-TDF.XML instances are not guaranteed to validate against child TDF schemas.

2.1 - TDF Structure

The BASE-TDF.XML specification has a consistent and simple concept of Assertions and Payloads. See [Figure 3](#). Assertions and Payloads are conceptual design elements that help explain the TDF structure. In the discussion that follows, physical TDF elements and attributes are preceded with the "tdf:" namespace prefix; this helps to distinguish the physical elements from the two conceptual design elements. For example, while "Assertion" refers to the conceptual design element, **tdf:Assertion** refers to the physical element.

There are two options for root elements: Trusted Data Object (TDO) and Trusted Data Collection (TDC). A TDO contains some data (the Payload) and some statements about that data (the Assertions). In the context of TDF, an 'Assertion' is defined as a statement providing handling, discovery, or mission metadata describing a Payload, TDO, or TDC, depending on the scope of the Assertion. A TDC contains a list of TDOs (the Payload) and some statements about those TDOs (the Assertions). A TDC may also be a collection of collections, and contain other TDCs. Without impacting access control, on rare occasions, HandlingAssertion types of data MAY be inserted into a regular **tdf:Assertion**.

Each TDO consists of one or more Assertions and a Payload. TDO Assertions may optionally be cryptographically bound to the Payload to provide assurance over the integrity of the Assertion(s), the Payload, and the relationship between the Assertion(s) and Payload.

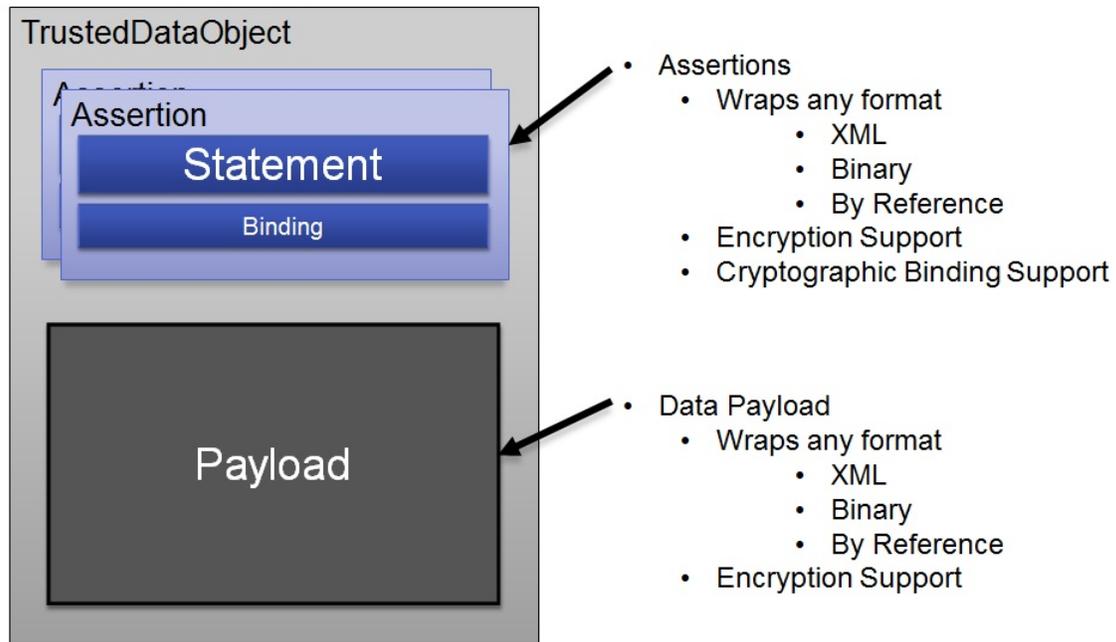


Figure 3 : Simple TDO

In a scenario where encryption is required, the TDO Assertion statements and/or TDO Payload may be optionally encrypted:

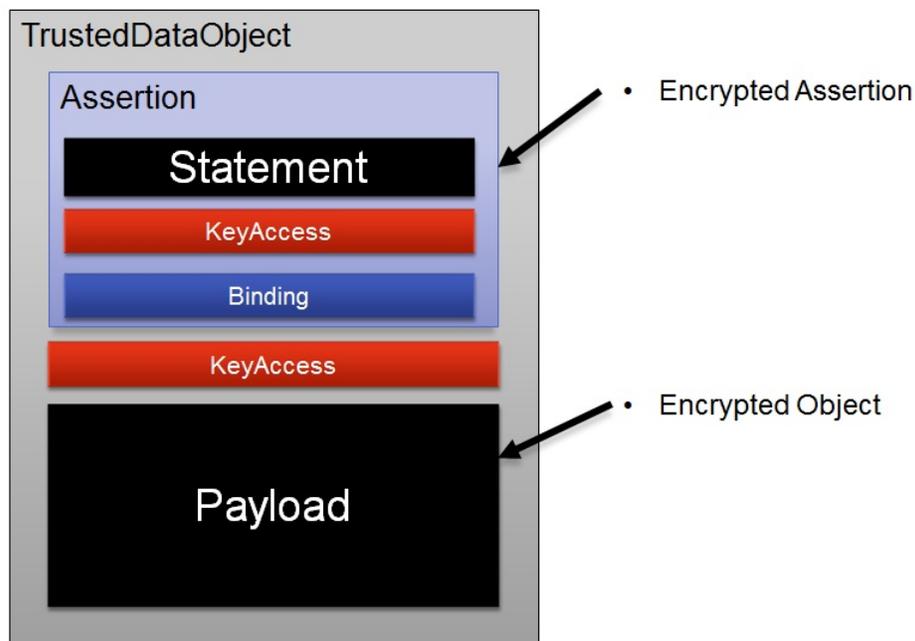


Figure 4 : TDO with Encryption

The Payload may be:

- Structured XML, implemented by the `tdf:StructuredPayload` element;
- String data, implemented by the `tdf:StringPayload` element;
- Base64Binary, implemented by the `tdf:Base64BinaryPayload` element; or
- A reference to an external Payload, implemented by the `tdf:ReferenceValuePayload` element.

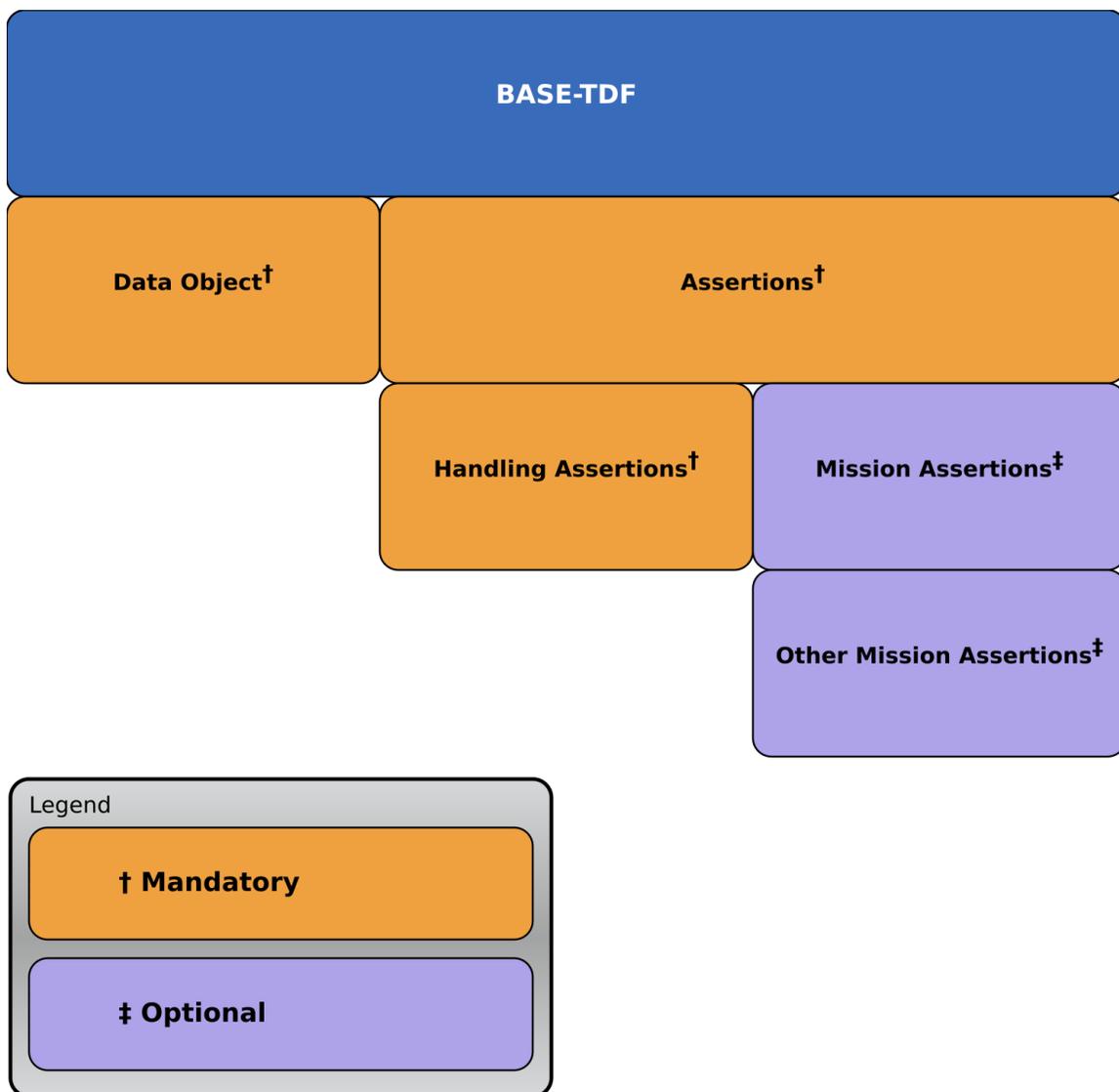


Figure 5 : TDF Structure

A TDC consists of a collection of TDOs or TDCs. It is expected, but not required, that the child TDOs and TDCs within a TDC are in some way related, with relationships encoded in the TDC Assertions. For example, in a biometric use case, a TDC might correspond to a biometric identity, with child TDOs corresponding to biometric modalities, such as finger prints, iris scans, and facial images. In this biometric use case, the root TDC `tdf:Assertion` elements would describe the

entire identity, while the child TDO `tdf:Assertion` elements would describe the individual modalities.

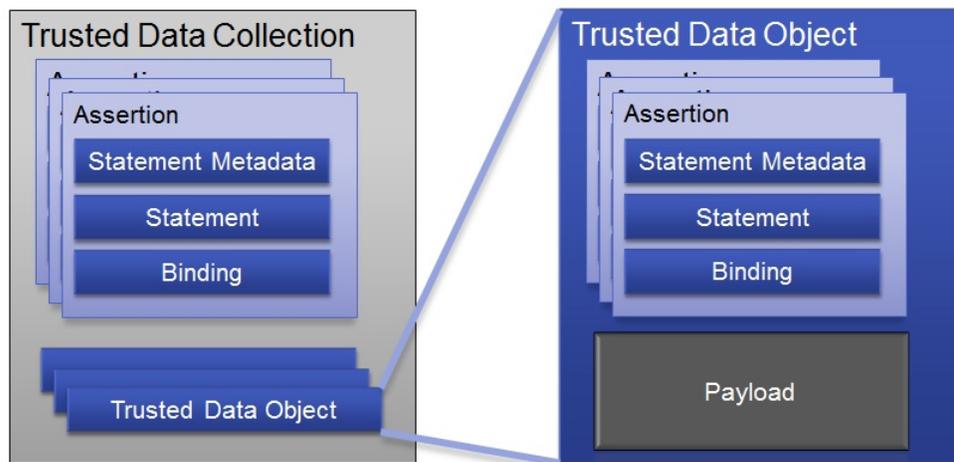


Figure 6 : Trusted Data Collection (TDC)

2.1.1 - Version Declarations

Specification versions are generally declared at the highest level of the XML structure that makes sense for its usage; generally either the root, or the first level of element that uses a specification. As such, many specifications used in a TDF are generally declared at the root.

As extension points, Assertions and Payloads may have different versions of specifications specified for use inside itself. However, it is also possible that an Assertion or Payload does not contain declarations for versions of specifications. In this case they are considered to be the same versions that are declared in the TDF. That is, the extension points inherit specification versions from the TDF in which they reside and, if they are extracted from the TDF, those version declarations **MUST** be copied into that content during extraction to maintain validity as well as comprehensibility.

2.2 - Assertions

2.2.1 - Assertion Scopes

Assertions can be scoped to apply to different portions of a BASE-TDF.XML instance. Several Assertion scopes imply certain meaning and processing instructions. TDF implements the concept of scope through the `@tdf:scope` attribute, which is used across all types of Assertions. The following sections explain the valid Assertion scopes for use within TDOs and TDCs and any additional processing requirements they imply.

2.2.1.1 - Assertion Scopes Within TDO

Assertions within a TDO can be scoped to apply to either the entire TDO, the Payload only, or both. The following tokens are used to specify the scope of Assertions within a TDO:

1. `@tdf:scope="PAYL"` means this Assertion applies only to the Payload within this TDO.
2. `@tdf:scope="TDO"` means this Assertion applies to every element within the TDO other than itself (includes peer `tdf:HandlingAssertions`, `tdf:Assertions`, and the Payload). This scope essentially means "the entire TDO".

2.2.1.2 - Assertion Scopes Within TDC

Assertions within a TDC can be scoped to apply to several different portions of a TDC instance. [Definition: The child TDOs and TDCs contained within a TDC are referred to as the *collection members*.]

[Definition: An Assertion with a *transitive scope* recursively applies to specific portions of each collection member within this TDC and MAY be inherited by a collection member if that collection member is extracted from this TDC.] Each transitive scope defines exactly which portions of the collection members the Assertion applies to and how the Assertion must be inherited when a collection member is extracted. Transitive scopes help reduce the need for duplicate Assertions within collection members. For example, instead of making an identical Assertion in each collection member individually, a single Assertion with a transitive scope at the TDC level may have the same intent with much less overhead.

[Definition: An Assertion with a *non-transitive scope* does not recursively apply to each collection member within this TDC and MUST NOT get inherited by a collection member if that collection member is extracted from this TDC.] Each non-transitive scope defines exactly which portion of this TDC the Assertion applies to. Non-transitive scopes are used for Assertions which only have meaning when considered in the scope of the TDC.

Whenever any change is made to the TDC, the intent of an Assertion may no longer logically apply depending upon the Assertion's scope and the change that was made. If a collection member is removed from the TDC, then the intent of an Assertion with a transitive scope still logically applies to the remaining subset of collection members. However, any other change made to the collection members within the TDC may logically invalidate an Assertion with a transitive scope (e.g., a new collection item is added or an existing collection member is modified). The intent of an Assertion with a non-transitive scope may no longer logically apply if any modification is made to the portions of the TDC to which the Assertion applies. Users modifying the TDC should understand the intent of each existing Assertion in order to correctly preserve their intent or make some corrective modification after changes have been made. [Section 2.3 - Binding and BindingInfo](#) outlines how to cryptographically bind an Assertion to the portions of the document to which it applies.

The following list defines the tokens used to specify the scope of Assertions within TDCs:

1. `@tdf:scope="TDC"` is a non-transitive scope and means this Assertion applies to all TDC elements collectively (other than itself). This includes peer `tdf:HandlingAssertion` elements, `tdf:Assertion` elements, `tdf:TrustedDataObject` elements, and

tdf:TrustedDataCollection elements. This scope essentially means "the entire TDC".

2. **@tdf:scope="DESC_TDO"** (short for descendant TDO) is a transitive scope and means this Assertion applies to every TDO contained within this TDC.

When a collection member is extracted from this TDC it MAY inherit Assertions with **@tdf:scope="DESC_TDO"** from its ancestor TDCs in the following ways:

If the collection member being extracted is a TDO, then any Assertion with **@tdf:scope="DESC_TDO"** in an ancestor TDC becomes an Assertion with **@tdf:scope="TDO"** in the extracted TDO.

If the collection member being extracted is a TDC, then any Assertion with **@tdf:scope="DESC_TDO"** in an ancestor TDC becomes an Assertion with **@tdf:scope="DESC_TDO"** in the extracted TDC.

3. **@tdf:scope="DESC_PAYL"** (short for descendant Payload) is a transitive scope and means this Assertion applies to every Payload within this TDC. This scope is similar to **"DESC_TDO"**, but this scope applies ONLY to the Payloads within descendent TDOs and does NOT include any **tdf:Assertion** or **tdf:HandlingAssertion** element of those TDOs.

When a collection member is extracted from this TDC it MAY inherit Assertions with **@tdf:scope="DESC_PAYL"** from its ancestor TDCs in the following ways:

If the collection member being extracted is a TDO, then any Assertion with **@tdf:scope="DESC_PAYL"** in an ancestor TDC becomes an Assertion with **@tdf:scope="PAYL"** in the extracted TDO.

If the collection member being extracted is a TDC, then any Assertion with **@tdf:scope="DESC_PAYL"** in an ancestor TDC becomes an Assertion with **@tdf:scope="DESC_PAYL"** in the extracted TDC.

4. **@tdf:scope="TDC_MEMBER"** is a non-transitive scope and means this Assertion applies to all collection members within this TDC. Unlike scope **"TDC"**, this scope does not apply to peer **tdf:HandlingAssertion** and **tdf:Assertion** elements.

This scope is useful for making an Assertion about the "current state" of the collection members within the TDC. For example, one might use the **"TDC_MEMBER"** scope to make an Assertion that all members of the TDC contain biometric modalities for a certain individual. However, as soon as any modification is made to the collection members, then the Assertion may no longer apply to the new state of the collection members (a collection member is added to the TDC, a collection member is removed from the TDC, any modification is made to any existing collection member).

2.2.1.3 - HandlingAssertion scopes within TDO

A TDO has at a minimum two **tdf:HandlingAssertion** elements: a TDO **tdf:HandlingAssertion** with **@tdf:scope="TDO"**, and a Payload **tdf:HandlingAssertion** with **@tdf:scope="PAYL"**. This allows for separate access

control decisions to be made for the Payload versus the entire TDO (which includes the Payload metadata). A `tdf:HandlingAssertion` MUST NOT be encrypted.

2.2.1.4 - HandlingAssertion scopes within TDC

A TDC can only have a single `tdf:HandlingAssertion` and its `@tdf:scope` must be "TDC". A `tdf:HandlingAssertion` MUST NOT be encrypted.

2.2.2 - Mission-Specific Metadata Assertions

Although missions may create their own unique set of Assertions, no understanding by the enterprise beyond access control is assured. The `tdf:Assertion/@tdf:type` attribute is intended to provide additional context allowing various systems to pre-determine relevance of Assertions without parsing or reading all of the Assertions. A `tdf:Assertion/@tdf:type` might include categorizations such as 'discovery', 'mission', or 'task order' to allow various systems to determine which Assertions are relevant for them to parse.

2.2.3 - Assertions and Data State

If an Assertion statement or a Payload is encrypted, then there are in fact two (potentially different) markings needed for decision making, analysis, and querying: one for describing the handling required for the ciphertext and the other for the handling required for the unencrypted (and in effect external) state. In cases where statements and/or Payloads are encrypted, `tdf:HandlingAssertion` elements indicate whether their marks apply to the ciphertext vs. plaintext by using the attribute `@tdf:appliesToState`.

The attribute `@tdf:appliesToState` can be used with `tdf:StatementMetadata` that is a child of `tdf:Assertion`. The `@tdf:appliesToState` attribute can only be used when content is encrypted, as indicated by the attribute `@tdf:isEncrypted`. When Payload content is encrypted (`@tdf:isEncrypted="true"`), it must be marked with two `tdf:HandlingAssertion` blocks, one indicating the classification and handling required for the cyphertext Payload (with `@tdf:appliesToState="encrypted"`), and the other indicating the classification and handling required for the plaintext Payload after decryption (with `@tdf:appliesToState="unencrypted"`). In this case, the `tdf:HandlingAssertion` that applies to the plaintext state is considered external to rollup, since the plain text content is not included in the instance. The `@tdf:appliesToState` attribute should only be used with `tdf:HandlingAssertion` elements scoped to the Payload.

2.3 - Binding and BindingInfo

A key concept in the TDF specification is the ability to cryptographically assure the relationship among portions of the document. This assurance is represented by the optional `tdf:Binding` element available on each `tdf:Assertion` and `tdf:HandlingAssertion`.

The `tdf:Binding` element includes information about the algorithm used to calculate the signature, the `tdf:signatureValue`.

In the current version of BASE-TDF.XML the **tdf:SignatureValue** is always calculated over a concatenation of the normalized portions of the document in the same order they appear in the document described by the **tdf:Assertion**.

The normalization method expressed in **tdf:Binding/tdf:SignatureValue/@tdf:normalizationMethod** is a Uniform Resource Identifier (URI) that provides guidance on how to format the included values such as whitespace, attributes, and child nodes in a universally consistent manner. The normalization method is essential to prevent formatting such as white-space and order from interfering with the validation of the cryptographic integrity of data. For example, XML canonicalization is one form of normalization that might be utilized. More information on XML canonicalization is available online at: [W3C Canonical XML](http://www.w3.org/TR/xml-c14n) [http://www.w3.org/TR/xml-c14n]. To use XML canonicalization as a normalization method, provide the URI to the form of XML canonicalization you are using, such as <http://www.w3.org/TR/2001/REC-xml-c14n-20010315> [http://www.w3.org/TR/2001/REC-xml-c14n-20010315] as the value for the **tdf:Binding/tdf:SignatureValue/@tdf:normalizationMethod**. This example URL is the URL defined in XML-SEC Rec for inclusive c14n without comments.

The expected portions of the document that each scope MUST include in the **tdf:SignatureValue** are detailed in the tables below. The abbreviation IFF stands for "if and only if". The pseudo XPathS in the tables below are not syntactically valid and use some abbreviations to save space and improve readability:

- Assume each element and attribute is in the BASE-TDF namespace.
- Payload refers to the *TDF extension points* **tdf:StringPayload**, **tdf:StructuredPayload**, **tdf:ReferenceValuePayload**, and **tdf:Base64BinaryPayload**.
- **tdf:AssertionStatement** refers to the *TDF extension points* **tdf:StringStatement**, **tdf:StructuredStatement**, **tdf:ReferenceStatement**, and **tdf:Base64BinaryStatement**.

Table 3 - TDO Binding Contents

XPath	Required to include in binding
tdf:TrustedDataObject/ tdf:Assertion[@tdf:scope='PAYL']	<ol style="list-style-type: none"> 1. ./tdf:AssertionStatement 2. ./tdf:StatementMetadata IFF ./tdf:Binding/tdf:SignatureValue/@tdf:includesStatementMetadata='true' 3. ../tdf:Payload
tdf:TrustedDataObject/ tdf:HandlingAssertion[@tdf:scope='PAYL']	<ol style="list-style-type: none"> 1. ./tdf:HandlingStatement 2. ../tdf:Payload

XPath	Required to include in binding
<p>tdf:TrustedDataObject/ tdf:Assertion[@tdf:scope='TDO']</p> <p>or</p> <p>tdf:TrustedDataObject/ tdf:HandlingAssertion[@tdf:scope='TDO']</p>	<ol style="list-style-type: none"> 1. ../tdf:HandlingAssertion/ tdf:HandlingStatement 2. ../tdf:Assertion/ tdf:AssertionStatement 3. ../tdf:Assertion/ tdf:StatementMetadata IFF ../ tdf:Binding/ tdf:SignatureValue/ @tdf:includesStatementMetadata ='true' 4. ../tdf:Payload

Table 4 - TDC Binding Contents

XPath	Required to include in binding
<p>tdf:TrustedDataCollection/ tdf:Assertion[@tdf:scope='TDC']</p> <p>or</p> <p>tdf:TrustedDataCollection/ tdf:HandlingAssertion[@tdf:scope='TDC']</p>	<ol style="list-style-type: none"> 1. ../tdf:HandlingAssertion/ tdf:HandlingStatement 2. ../tdf:Assertion/ tdf:AssertionStatement 3. ../tdf:Assertion/ tdf:StatementMetadata IFF ../ tdf:Binding/ tdf:SignatureValue/ @tdf:includesStatementMetadata ='true' 4. ../tdf:TrustedDataObject/ tdf:HandlingAssertion/ tdf:HandlingStatement 5. ../tdf:TrustedDataObject/ tdf:Assertion/ tdf:AssertionStatement 6. ../tdf:TrustedDataObject/ tdf:Assertion/ tdf:StatementMetadata IFF ../ tdf:Binding/ tdf:SignatureValue/ @tdf:includesStatementMetadata ='true' 7. ../tdf:TrustedDataObject/ tdf:Payload 8. ../tdf:TrustedDataCollection/ tdf:HandlingAssertion/ tdf:HandlingStatement 9. ../tdf:TrustedDataCollection/ tdf:Assertion/ tdf:AssertionStatement 10. ../tdf:TrustedDataCollection/ tdf:Assertion/ tdf:StatementMetadata IFF ../ tdf:Binding/ tdf:SignatureValue/ @tdf:includesStatementMetadata ='true'

XPath	Required to include in binding
<p>tdf:TrustedDataCollection/ tdf:Assertion[@tdf:scope='DESC_TDO']]</p> <p>or</p> <p>tdf:TrustedDataCollection/ tdf:Assertion[@tdf:scope='TDC_MEMBER']</p>	<ol style="list-style-type: none"> 1. ./tdf:AssertionStatement 2. ./tdf:StatementMetadata IFF ./tdf:Binding/ tdf:SignatureValue/ @tdf:includesStatementMetadata='true' 3. ../tdf:TrustedDataObject/ tdf:HandlingAssertion/ tdf:HandlingStatement 4. ../tdf:TrustedDataObject/ tdf:Assertion/ tdf:AssertionStatement 5. ../tdf:TrustedDataObject/ tdf:Assertion/ tdf:StatementMetadata IFF ./tdf:Binding/ tdf:SignatureValue/ @tdf:includesStatementMetadata='true' 6. ../tdf:TrustedDataObject/ tdf:Payload 7. ../tdf:TrustedDataCollection/ tdf:HandlingAssertion/ tdf:HandlingStatement 8. ../tdf:TrustedDataCollection/ tdf:Assertion/ tdf:AssertionStatement 9. ../tdf:TrustedDataCollection/ tdf:Assertion/ tdf:StatementMetadata IFF ./tdf:Binding/ tdf:SignatureValue/ @tdf:includesStatementMetadata='true'

XPath	Required to include in binding
tdf:TrustedDataCollection/ tdf:Assertion[@tdf:scope='DESC_PAY L']	<ol style="list-style-type: none"> 1. ./tdf:AssertionStatement 2. ./tdf:StatementMetadata IFF ./tdf:Binding/ tdf:SignatureValue/ @tdf:includesStatementMetadata='true' 3. ../tdf:TrustedDataObject/ tdf:Payload

2.4 - Normalization Method

The normalization method expressed in `tdf:Binding/tdf:SignatureValue/@tdf:normalizationMethod` and `tdf:Binding/tdf:BoundValueList/tdf:BoundValue/@tdf:normalizationMethod` is a URI that provides guidance on how to format the included values such as whitespace, attributes, and child nodes in a universally consistent manner. The normalization method is essential to prevent formatting such as whitespace and order from interfering with the validation of the cryptographic integrity of data. For example, XML canonicalization is one form of normalization that might be utilized. The table below lists several XML canonicalization URLs.

Table 5 - Sample URLs for XML Canonicalization Normalization Methods

Sample NormalizationMethod URL	Description
http://www.w3.org/TR/2001/REC-xml-c14n-20010315	The URL defined in XML-SEC Rec for inclusive c14n without comments.
http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments	The URL defined in XML-SEC Rec for inclusive c14n with comments.
http://www.w3.org/2001/10/xml-exc-c14n#	The URL defined in XML-SEC Rec for exclusive c14n without comments.
http://www.w3.org/2001/10/xml-exc-c14n#WithComments	The URL defined in XML-SEC Rec for exclusive c14n with comments.
http://www.w3.org/TR/xml-c14n1	The URI for inclusive c14n 1.1 without comments.
http://www.w3.org/TR/xml-c14n1#WithComments	The URI for inclusive c14n 1.1 with comments.

2.5 - Encryption and EncryptionInfo

A key concept in the TDF specification is the ability to encrypt Payloads, Assertions, and keys. Whenever content is encrypted, encryption information must be provided.

`tdf:EncryptionInformation` contains `tdf:KeyAccess` and `tdf:EncryptionMethod` information, providing the information necessary for decryption or key retrieval. Onion or layered encryption is also supported. In this case, there will be multiple `tdf:EncryptionInformation`

elements within one `tdf:EncryptionInformation` group. Each `tdf:EncryptionInformation` has an optional `@tdf:sequenceNum` attribute that is required to be provided when multiple `tdf:EncryptionInformation` elements are used. The order of sequence for encryption should be in increasing numerical order. The highest `@tdf:sequenceNum` value corresponds to the outermost layer of encryption. For example, this layered or onion encryption may be required in a use case where both a system and a user must provide certificates before information can be decrypted. Encryption Method allows key size, algorithm, and OAEP, *Optimal Asymmetric Encryption Padding Scheme* [\[11\]](#) information.

2.6 - Linked or Embedded Data Objects

Linked objects classification does NOT impact the classification of the TDO. Embedded objects classification does impact the classification of the TDO.

2.7 - MIME type

The Media Type (MIME) type for a BASE-TDF.XML document is `application/dni-tdf+xml`. This is a convention for our community. This type has NOT been registered with the Internet Assigned Numbers Authority (IANA). Should there be a conflict in the future it will be addressed at that time. Systems can use this MIME type to facilitate communications and address business needs within the community.

2.8 - Payload Chunking

For reference value payloads that may need to pass through a transport mechanism, network boundary, or cross-domain guard that does not support large single files, the capability to split payloads into multiple smaller files, also known as "chunking" is supported in `tdf:ReferenceValuePayload`.

2.9 - BASE-TDF Schematron Rules

BASE-TDF.XML schematron rules should be used for validation in derived child TDF instances along with the child TDF schematron rules.

Chapter 3 - Constraints

3.1 - Data Validation Constraint Rules

3.1.1 - Purpose

The BASE-TDF.XML schema defines the data elements, attributes, cardinalities and parent-child relationships for which XML instances must comply. Validation of these syntax aspects is an important first step in the validation process. An additional level of validation is needed to ensure that the content complies with the constraints as specified in applicable IC policy guidance and codified in these constraint rules. Traditional schema languages are generally unable to effectively represent these additional constraints. For more information, please see the “Data Validation Constraint Rules” chapter in the IC-SF.XML^[2] framework document.

3.1.2 - Value Enumeration Constraints

Several elements and attributes of the BASE-TDF.XML model use Controlled Vocabulary Enumeration (CVE)s to define the data allowed in the element or attribute. In some cases the specific CVE is specified via an attribute, which may include a default CVE. Further, in some of the cases where the CVE can be specified, the attribute may restrict the list of CVEs allowed and some may allow for the author to specify their own CVE. For each of these, the value must be in the specified external CVE or the default CVE.

Some CVEs are not available on all networks. A subset CVE will be provided for use on networks not approved for the entire list. If the processing will occur on a network where the entire CVE is not available, the subset CVE may be substituted in the constraint rules since the excluded values would be excluded from use on the lower network.

As noted in the specific rules, a failure of validation against a CVE will generate an Error.

3.1.3 - Additional Constraints

3.1.3.1 - DES Constraints

The DES version is specified through attributes on the root element. The schema constrains the values of these attributes. The `@DESVersion` attribute enables systems processing an instance document to be certain which set of constraint rules, schema, CVEs and business rules are intended by the author to be used.

3.1.4 - Constraint Rules

The detailed constraint rules for the BASE-TDF.XML schema can be found in a separate document inside the Documents/BASE-TDF directory, in the “BASE-TDF_Rules.pdf” file. This document is generated from the individual Schematron files to provide a single searchable document for all of the constraint rules encoded in Schematron. Obsolete rule numbers are listed in the “BASE-TDF_Rules.pdf” file.

3.2 - Data Rendering Constraint Rules

3.2.1 - Purpose

Rendering rules define constraints on the rendering and display of BASE-TDF.XML documents. The intent is to inform the development of systems capable of rendering or displaying BASE-TDF.XML data for use by individuals not familiar with the details of the BASE-TDF.XML markup. While expressed in a similar manner to the data validation constraint rules above, there is no expectation that evaluation of these rules can be automated; rather these rules should inform the evaluation of a system's capabilities and functionality.

3.2.2 - Rendering Constraint Rules

The following table contains the information for the BASE-TDF.XML data rendering constraint rules.

Table 6 - Constraint Rules

Rule Number	Severity	Description	Human Readable Description
There are no Data Rendering Constraint rules at this time.			

Chapter 4 - Conformance Validation

An instance is considered conformant with the BASE-TDF.XML specification if it passes all of the following normative validation steps. The following steps do not dictate how this validation strategy is implemented.

4.1 - Definitions

Terms are defined the first time they are used. Definitions are cumulative, meaning that a term used in any given step may be defined in a previous step. The following definitions are global concepts, so they are defined in this section instead of in-line.

[Definition: A *TDF extension point* is an element within the BASE-TDF.XML specification whose purpose is to hold multiple forms of user content in-line.] There are six extension points within BASE-TDF.XML:

1. `tdf:StringStatement`
2. `tdf:Base64BinaryStatement`
3. `tdf:StructuredStatement`
4. `tdf:StringPayload`
5. `tdf:Base64BinaryPayload`
6. `tdf:StructuredPayload`.

Note that `tdf:ReferenceStatement` and `tdf:ReferenceValuePayload` are not considered extension points because they only convey a link to content and do not hold content in-line. `tdf:HashVerification` contains hash verification information with regards to the referenced statement or payload and is not referring to the hash verification of any intermediate URI redirects that may exist.

[Definition: The content contained within elements `tdf:Base64BinaryStatement` and `tdf:Base64BinaryPayload` is referred to as *binary content*.]

[Definition: The content contained within elements `tdf:StringStatement` and `tdf:StringPayload` is referred to as *string content*.]

[Definition: The content contained within elements `tdf:StructuredStatement` and `tdf:StructuredPayload` is referred to as *structured content*.]

[Definition: The term *TDO structure* refers to all elements within an BASE-TDF.XML instance excluding the content of any TDF extension point.]

4.2 - Why a verbose validation strategy is required

The BASE-TDF.XML specification is designed to be extremely flexible by allowing users to include several formats of in-line content in several extension points (see [Figure 7](#)). These *TDF extension*

points require BASE-TDF.XML instances to use a more verbose validation strategy for several reasons:

1. BASE-TDF.XML schema defines the extension point `tdf:StructuredPayload` as `@xs:any processContents="skip/"`, which skips all schema validation for the content contained within those extension points.
2. *Structured content* within the BASE-TDF.XML instance can contain data which can conflict with the data contained within the elements declared as part of the BASE-TDF.XML specification.
3. For *binary content* and *string content*, XSD schema validation and XML business rules are not applicable and custom validation logic is required to validate that content.



Figure 7 : TDF Extension Points

4.3 - Required Order of HandlingAssertions

Before any validation takes place on a TDO, a validation implementation MUST ensure that the TDO `tdf:HandlingAssertion` is the first `tdf:HandlingAssertion` in document order.

[Definition: The `tdf:HandlingAssertion` element which specifies attribute `@tdf:scope` with a value containing "TDC" is referred to as the *TDC tdf:HandlingAssertion* .]

Before any validation takes place on a TDC, a validation implementation MUST ensure that the TDC `tdf:HandlingAssertion` is the first `tdf:HandlingAssertion` in document order.

The banner level markings within an BASE-TDF.XML instance are contained within a `tdf:HandlingAssertion` element and an instance may have multiple `tdf:HandlingAssertion` elements, each specifying a different scope. It is required that the first `tdf:HandlingAssertion` element in document order contain the banner level markings intended for the entire BASE-TDF.XML instance.

[Definition: The `tdf:HandlingAssertion` element which specifies attribute `@tdf:scope` with a value containing "PAYL" is referred to as the Payload `tdf:HandlingAssertion` .] [Definition: The `tdf:HandlingAssertion` element which specifies attribute `@tdf:scope` with a value containing "TDO" is referred to as the *TDO tdf:HandlingAssertion* .]

4.4 - TDO Validation Steps

This section outlines the required steps to fully validate a TrustedDataObject (TDO).

4.4.1 - Step 1 - TDO aware and cross Assertion constraints

This step is intended to support validation which requires knowledge of the TDO structure.

BASE-TDF.XML validation, to include schema and business rules, should be run during this step.

TDO aware validation MAY be performed during this step. For example, one might want to run business rules specific to a certain domain or system. Some examples of custom validation could include:

- If this TDO contains an Assertion with child element X, then it must also contain a peer Assertion with child element Y.
- Verify that this TDO instance contains a custom Assertion specific to a certain domain.
- Verify all bindings within this TDO.
- If the Payload is encrypted, attempt to decrypt it and run additional custom validation on the decrypted content.

4.4.2 - Step 2 – Extension point constraints

This step is intended to support validation for the content of all *TDF extension points* contained within the TDO.

The child content of any *TDF extension point* MAY be validated. Any content validated in this step MUST be validated independently and in isolation. Determining which *TDF extension points* are validated in this step is implementation specific. For example, an implementation might choose to only validate *structured content* while ignoring *binary content* and *string content* completely. Or, an

implementation might define a configuration which only validates *structured content* whose root element is in a certain namespace or set of namespaces.

4.5 - TDC Validation Steps

This section outlines the required steps to fully validate a TrustedDataCollection (TDC).

4.5.1 - Step 1 – TDC aware and cross Assertion constraints

This step is intended to support validation which requires knowledge of the TDC structure.

BASE-TDF.XML validation to include schema and business rules should be run during this step.

Additional validation may be performed during this step. For example, one might want to run business rules specific to a certain domain or system. Some examples of custom validation could include:

- Test if this TDC contains an Assertion with child element X, then it must also contain a peer Assertion with child element Y.
- Test if this TDC must contain a certain Assertion type, such as a Multi Audience Collection (MAC) Assertion.

4.5.2 - Step 2 – Extension point constraints

This step is intended to support validation for the TDF extension point content contained within child `tdf:Assertion` elements of the TDC. The rules outlined in [Section 4.4.2 - Step 2 - Extension point constraints](#) should be applied to each child `tdf:Assertion` element of the `tdf:TrustedDataCollection` element.

4.5.3 - Step 3 - Recursive Validation

A `tdf:TrustedDataCollection` element supports recursion by allowing child `tdf:TrustedDataObject` and `tdf:TrustedDataCollection` elements. Each `tdf:TrustedDataObject` element must be validated according to the steps outlined in [Section 4.4 - TDO Validation Steps](#). Each `tdf:TrustedDataCollection` element must be validated according to the steps outlined in [Section 4.5 - TDC Validation Steps](#).

Chapter 5 - Future Features

5.1 - Explicit Scope

In future versions, the concept of scope will be extended to support a flexible, explicit list of elements. The token "**EXPLICIT**" is expected to be used to indicate this granularity. An Assertion using explicit scope will require either a **tdf:ReferenceList** or a **tdf:BoundValueList** and the elements to which it "applies" will be determined by the values in the **tdf:ReferenceList** or **tdf:BoundValueList**.

5.2 - BoundValueList

A key concept in the TDF specification is the ability to cryptographically assure the relationship among portions of the document. Future versions of TDF will make Cryptographic Binding more flexible and granular through the introduction of an optional Bound Value List as a child of the **tdf:Binding** element. A **tdf:BoundValueList** is a container of bound value references that point to the elements that are included in a cryptographic binding. The **@idref** attribute of **tdf:BoundValue** or **tdf:Reference** element is the internal instance reference to the element being bound. The intent of the **tdf:BoundValueList** is to allow granular control over the scope of the binding signature. In the future, when **tdf:BoundValueList** is present, the **tdf:SignatureValue** will be calculated over the normalized value of the **tdf:BoundValueList** using the normalization method denoted in the **tdf:Binding/tdf:SignatureValue/@tdf:normalizationMethod** attribute.

In BASE-TDF.XML, where the **tdf:BoundValueList** is not present, the **tdf:signatureValue** is always calculated over a concatenation of the normalized portions of the document in the same order they appear in the document described by the Assertion.

The normalization method expressed in **tdf:Binding/tdf:SignatureValue/@tdf:normalizationMethod** and **tdf:Binding/tdf:BoundValueList/tdf:BoundValue/@tdf:normalizationMethod** is a URI that provides guidance on how to format the included values such as whitespace, attributes, and child nodes in a universally consistent manner. The normalization method is essential to prevent formatting such as white-space and order from interfering with the validation of the cryptographic integrity of data. For example, XML canonicalization is one form of normalization that might be utilized. More information on XML canonicalization is available online at: [W3C Canonical XML](http://www.w3.org/TR/xml-c14n) [http://www.w3.org/TR/xml-c14n]. To use XML canonicalization as a normalization method, provide the URI to the form of XML canonicalization you are using, such as <http://www.w3.org/TR/2001/REC-xml-c14n-20010315> [http://www.w3.org/TR/2001/REC-xml-c14n-20010315] as the value for the **tdf:Binding/tdf:SignatureValue/@tdf:normalizationMethod**. This example URL is the URL defined in XML-SEC Rec for inclusive c14n without comments.

Appendix A Feature Summary

The following tables summarize major features by version for BASE-TDF.XML. The “Required date” is the date when systems SHOULD support a feature based on the specified driver. Executive Orders, Information Security Oversight Office (ISOO) notices, ICDs and other policy documents have a variety of effective dates. The “Required date” may be later than the date of applicable policy based on the effective date defined in the policy (e.g., The IC Marking System Register and Manual^[1] has an implementation date of one year after issuance).

Table 7 - Feature Summary Legend

Key	Description
F	Full (able to comply and verified by spec to some degree)
P	Partial (Able to comply but not verifiable)
N	Non-compliance (Can't comply)
N/A	Not Applicable. Feature is no longer required.
Cell Colors represent the same information as the Key value	

A.1. BASE-TDF Feature Summary

Table 8 - BASE-TDF Feature comparison

Required date	Feature	V2021-JAN	V2021-NOV
	Base TDF structure for use with all other TDF specifications.	F	F
	Allow hash elements in ReferenceValueType independent of blocking.	N	F

Appendix B Change History

The following table summarizes the version identifier history for this DES.

Table 9 - DES Version Identifier History

Version	Date	Purpose
2021-JAN	January 15, 2021	Initial Release. For details, see Section B.2 - V2021-JAN Initial Release Summary
2021-NOV	December 3, 2021	Routine revision to technical specification. For details of changes, see Section B.1 - V2021-NOV Release Summary

B.1 - V2021-NOV Release Summary

Significant drivers for Version V2021-NOV include:

- Community Change Requests

The following table summarizes the changes made to 2021-JAN in developing 2021-NOV.

Table 10 - Data Encoding Specification V2021-NOV Initial Release Summary

#	Change	Artifacts changed	Compatibility Notes
1	Added rule to check for existence of optional sf:DESVersion attribute when sf:hashv elements or attributes are used. (CR-2021-008)	Documentation Schematron BASE-TDF-ID-00022 added	Data generation and ingestion systems need to be updated.
2	Fix bug in ReferenceValueType to allow hashes without chunking and added rule to give warning if binding signature exists but no hash is provided. (CR-2021-008)	Documentation Schema Schematron BASE-TDF-ID-00023 added BASE-TDF-ID-00024 added	Data generation and ingestion systems need to be updated.

B.2 - V2021-JAN Initial Release Summary

Significant drivers for Version V2021-JAN include:

- Creation of BASE-TDF.XML specification.

The following table summarizes the initial release in V2021-JAN.

Table 11 - Data Encoding Specification V2021-JAN Initial Release Summary

#	Change	Artifacts changed	Compatibility Notes
1	Creation of BASE-TDF.XML specification.(CR-2019-057, CR-2020-034, CR-2020-039)	Documentation CVE Schema Schematron XSL	Initial Release.

Appendix C List of Abbreviations

This appendix lists all the acronyms and abbreviations referenced in this encoding specification.

CVE	Controlled Vocabulary Enumeration
DES	Data Encoding Specification
DNI	Director of National Intelligence
IANA	Internet Assigned Numbers Authority
IC	Intelligence Community
IC CIO	Intelligence Community Chief Information Officer
ICD	Intelligence Community Directive
IC ESB	Intelligence Community Enterprise Standards Baseline
ICPM	Intelligence Community Policy Memorandum
ICS	Intelligence Community Standard
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
ISOO	Information Security Oversight Office
MAC	Multi Audience Collection
MIME	Media Type
TDC	Trusted Data Collection
TDF	Trusted Data Format
TDO	Trusted Data Object
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	XSL Transformations

Appendix D Bibliography

[1] IC Markings

Director of National Intelligence (DNI), Special Security Directorate (SSD), Security Markings Program (SMP). *Intelligence Community Markings System Register and Manual*. Available online Intelink-TS at: <https://go.ic.gov/tGXkwGO> (case sensitive – tango Golf Xray kilo whiskey Golf Oscar)
Available online Intelink-U at: <https://w3id.org/ic/standards/policy/icmarkings>

[2] IC-SF.XML

Office of the Director of National Intelligence. *Intelligence Community Specification Framework (IC-SF.XML)*. Available online Intelink-TS at: <https://go.ic.gov/pNFyuVg> (case sensitive – papa November Foxtrot yankee uniform Victor golf)
Available online Intelink-U at: <https://w3id.org/ic/standards/IC-SF>
Available online at: <https://w3id.org/ic/standards/public>

[3] IC-TDF.XML

Office of the Director of National Intelligence. *XML Data Encoding Specification for Trusted Data Format (IC-TDF.XML)*. Available online Intelink-TS at: <https://go.ic.gov/hdwc8fn> (case sensitive – hotel delta whiskey charlie 8 foxtrot november)
Available online Intelink-U at: <https://w3id.org/ic/standards/TDF>
Available online at: <https://w3id.org/ic/standards/public>

[4] ICD 208

Office of the Director of National Intelligence. *Write For Maximum Utility*. Intelligence Community Directive 208. 17 December 2008.
Available online at: http://www.dni.gov/files/documents/ICD/icd_208.pdf

[5] ICD 209

Office of the Director of National Intelligence. *Tearline Production and Dissemination*. Intelligence Community Directive 209. 6 September 2012.
Available online at: <http://www.dni.gov/files/documents/ICD/ICD%20209%20Tearline%20Production%20and%20Dissemination.pdf>

[6] ICD 500

Office of the Director of National Intelligence. *Director of National Intelligence Chief Information Officer*. Intelligence Community Directive 500. 7 August 2008.
Available online Intelink-TS at: <https://go.ic.gov/U7v6ZRL> (case sensitive – Uniform 7 victor 6 Zulu Romeo Lima)
Available online at: http://www.dni.gov/files/documents/ICD/ICD_500.pdf

[7] ICD 501

Office of the Director of National Intelligence. *Discovery and Dissemination or Retrieval of Information within the Intelligence Community*. Intelligence Community Directive 501. 21 January 2009.
Available online Intelink-TS at: <https://go.ic.gov/FTBM8OS> (case sensitive – foxtrot Tango Bravo Mike 8 Oscar Sierra)

Available online at: http://www.dni.gov/files/documents/ICD/ICD_501.pdf

[8] ICPM 2007-200-2

Office of the Director of National Intelligence. *Preparing Intelligence to Meet the Intelligence Community's Responsibility to Provide*. Intelligence Community Policy Memorandum 2007-200-2. 11 December 2007.

Available online at: <http://www.dni.gov/files/documents/IC%20Policy%20Memos/ICPM%202007-200-2%20Responsibility%20to%20Provide.pdf>

[9] ICS 500-20

Director of National Intelligence Chief Information Officer. *Intelligence Community Enterprise Standards Compliance*. Intelligence Community Standard 500-20. 16 December 2010.

Available online Intelink-TS at: <https://go.ic.gov/kh8NMVJ> (case sensitive – kilo hotel 8 November Mike Victor Juliet)

Available online Intelink-U at: <https://w3id.org/ic/standards/policy/ICS500-20>

[10] ICS 500-21

Director of National Intelligence Chief Information Officer. *Tagging of Intelligence and Intelligence-Related Information*. Intelligence Community Standard 500-21. 28 January 2011.

Available online Intelink-TS at: <https://go.ic.gov/0Agmenr> (case sensitive – 0 Alpha golf mike echo november romeo)

Available online Intelink-U at: <https://w3id.org/ic/standards/policy/ICS500-21>

[11] OAEP

Mihir Bellare, Phillip Rogaway. *Optimal Asymmetric Encryption Padding Scheme (OAEP)*.

Available for purchase at: <http://dx.doi.org/10.1007/BFb0053428>

Conference online at: <http://www.informatik.uni-trier.de/~ley/db/conf/eurocrypt/eurocrypt94.html>

[12] Schematron

International Organization for Standardization (ISO). *Information technology -- Document Schema Definition Language (DSDL) -- Part 3: Rule-based validation -- Schematron*. ISO/IEC 19757-3:2006.

ISO Spec Available online at: <http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html>

StyleSheets for compiling Available online at: <http://code.google.com/p/schematron/>

[13] XSLT2

World Wide Web Consortium (W3C). *XSL Transformations (XSLT) Version 2.0*. W3C Recommendation 23 January 2007.

Available online at: <http://www.w3.org/TR/xslt20/>

Appendix E Points of Contact

The Intelligence Community Chief Information Officer (IC CIO) facilitates one or more collaboration and coordination forums charged with the adoption, modification, development, and governance of IC technical specifications of common concern. This technical specification was produced by the IC CIO and coordinated with these forums, approved by the IC CIO or a designated representative, and made available at the following Director of National Intelligence (DNI)-sponsored web sites.

Public Website: <https://w3id.org/ic/standards/public>

Intelshare: <https://w3id.org/ic/standards/data-specs>

Direct all inquiries about this IC technical specification, IC technical specification collaboration and coordination forums, or IC element representatives involved in those forums, to the IC CIO.

E-mail: ic-standards-support@odni.gov.

Appendix F IC CIO Approval Memo

An IC CIO Approval Memo should accompany this enterprise technical data specification bearing the signature of the IC CIO or an IC CIO-designated official(s). If an IC CIO Approval Memo is not accompanying this specification's version release package, then refer back to the authoritative web location(s) for this specification to see if a more complete package or a specification update is available.

Specification artifacts display a date representing the last time a version's artifacts as a whole were modified. This date most often represents the conclusion of the IC Element collaboration and coordination process. Once the IC Element coordination process is complete, the specification goes through an internal IC CIO staffing and coordination process leading to signature of the IC CIO Approval Memo. The signature date of the IC CIO Approval Memo will be later than the last modified date shown on the specification artifacts by an indeterminable time period.

Upon signature of the IC CIO Approval Memo, IC Elements may begin to use this specification version in order to address mission and business objectives. However, it is critical for IC Elements, prior to disseminating information encoded with this new specification version, to ensure that key enterprise services and consumers are prepared to accept this information. IC Elements should work with enterprise service providers and consumers to orchestrate an orderly implementation transition to this specification version in concert with mandatory and retirement usage decisions captured in the Intelligence Community Enterprise Standards Baseline (IC ESB) as defined in ICS 500-20^[9].