IEEE P1484.2/D7, 2000-11-28
Draft Standard for Learning Technology — Public and Private Information (PAPI) for Learners (PAPI Learner)

Sponsored by the Learning Technology Standards Committee
of the IEEE Computer Society

[Note: Information about IEEE LTSC P1484.2 can be found at:

`http://ieee.ltsc.org/wg2`

This document (PAPI Learner draft 6) is also available at:

`http://edutool.com/papi`

This note will be removed upon reaching the final draft of this IEEE document.]

Introduction

(This introduction is not part of IEEE P1484.2, Public and Private Information (PAPI) for Learners.)

** TO BE SUPPLIED **

At the time this Standard was completed, the working group had the following membership:

Mike Collett, *Chair*
Frank Farance, *Technical Editor*

aaa
Carlos. C Amaro
Thor Anderson
Debbie Brown
Peter Brusilovsky
Richard Burke
Brant Cheikes
Philip Dodds
Mike Fore
Paul Foster
Vladimir Goodkodsky
Martha Gray
Wayne Hodgins
Roger Lange
Ian Kegel
Cindy Mazow
William A. McDonald
Bill Melton
Yves Nicol
Claude Ostyn
Bruce Peoples
Tom Probert
Dan Rehak
Kevin Riley
Steve Ritter
Robby Robson
Randy Saunders
Jim Schoening
Paul Siegel
Kathy Sinitsa
Gayle Stroup
Josh Tonkel
Brendon Towle
John Tyler
Tom Wason
Eamonn Webster
Ian Wright
Bill Young
zzz
To be supplied

The following persons were on the balloting committee: (To be provided by IEEE editor at time of publication.)

CONTENTS

1 Overview

Abstract

The Public and Private Information (PAPI) for Learners (PAPI Learner) is a standard "portable" learner (student) records. PAPI Learner is a data interchange specification, i.e., communication among cooperating systems. The data is exchanged: (1) via external specification, i.e., only PAPI Learner coding bindings are used while some other data communication method is mutually agreed upon by data exchange participants; (2) via control transfer mechanism to facilitate data interchange, e.g., PAPI Learner API bindings; (3) via data and control transfer mechanisms. e.g., PAPI Learner protocol bindings.

A key feature of the PAPI Learner Standard is the logical division, separate security, and separate administration of several types of learner information: (1) personal information. e.g., name, address, social security number; (2) relations information, e.g., cohorts, classmates, (3) security information, e.g., public keys, private keys, credentials; (4) preference information, e.g., useful and unusable I/O devices, learning styles, physical limitations; (5) performance information, e.g., grades, interim reports, log books; (6) portfolio information, e.g., accomplishments and works. These six types of information are also known as "profile information" and "learner profiles". The PAPI Learner Standard may be integrated with other systems, protocols, formats, and technologies.

1.1 Scope

This Standard specifies the syntax and semantics of a "Learner Model", which characterizes a learner (student or knowledge worker) and his or her knowledge/abilities. This Standard includes elements for recording knowledge acquisition, skills, abilities, learning styles, records, and personal information. This Standard allows these elements to be represented in multiple levels of granularity, from a coarse overview, down to the smallest conceivable sub-element. The Standard allows different views of the Learner Model (learner, teacher, parent, school, employer, etc.) and substantially addresses issues of privacy and security.

Note: Initially, this Standard was developed for learning technology applications but the PAPI approach may be easily applied to other types of human-related information such as medical and financial applications.

The following features are *outside* the scope of this Standard:

- **Specific Extensions.** Not all human information about learners is specified by this Standard. For example, "personal information", in general, might include "shoe size", but the PAPI learner personal information is a limited subset of "personal information", which *does not* require the inclusion of "shoe size". Implementations may provide their own data extensions to this Standard, but these systems might change from "strictly conforming implementations" to "conforming implementations". *This Standard supports data extension mechanisms that may be used by users, groups, vendors, institutions, industries, and others.*

- **Granularity.** This Standard does not specify the granularity of information. For example, in learning technology applications PAPI learner records can have granularity ranging from professional certifications (e.g., years of learning), to semester grades (e.g., months of learning), to lesson scores (e.g., days of learning), to minute-by-minute data samples of learner progress (e.g., minutes of learning). *This Standard may be used for a wide range of data recording applications.*
- **Repository Design.** This Standard does not specify the implementation or administration of the records repositories but supports a wide variety of design and implementation possibilities. For example, all types of PAPI learner information may be implemented as a single, combined repository or may be implemented as separate repositories. *The specific design of the repository (or repositories) is a "quality of implementation" feature and is outside the scope of this Standard. The choice and design of partitioning repositories into one or more administrative "realms" is a feature of the implementation and application, and is outside the scope of this Standard.*
- **Specific Security Technologies.** This Standard supports the incorporation of various security architectures, methods, and techniques that support a wide range of implementations and security policies. *However, no specific security technologies (e.g., 128-bit encryption) are mandated by this Standard.*

1.2 Purpose

The purpose of this Standard is:

- To enable learners (students or knowledge workers) of any age, background, location, means, or school/work situation to create and build a personal Learner Model, based on standards, which they can utilize throughout their education, learning experiences, and work life.
- To enable courseware developers to develop materials that will provide more personalized and effective instruction.
- To provide educational researchers with a standardized and growing source of data.
- To provide a foundation for the development of additional educational standards, and to do so from a student-centered learning focus.
- To provide architectural guidance to education system designers.

This Standard describes interoperability that is:

- **General.** This Standard may be implemented across a wide variety of platforms, operating environments, applications, and industries.
- **Lasting.** This Standard is expected to span a technical horizon of 5-10 years. Conforming implementations are expected to be interoperable for at least the duration of the technical horizon. Technical Corrigenda (corrections), Amendments (enhancements), and Revisions (regular 5-year cycle) will be developed through the accredited standards development process to assure continuity and consistency for stakeholders.
- **Precise.** Interoperability can be tested, measured, and assessed for implementations that claim conformance to this Standard.
- **Formal.** The IEEE 1484.2 Learner Model Working Group is responsible for: developing this Standard; maintaining this Standard; correcting defects within this Standard; and for formal interpretation of this Standard resolution of any ambiguities.

IEEE is an open, accredited standards development organization that administers the formal standards process for the Learning Technology Standards Committee (IEEE LTSC) and IEEE 1484.2, a working group within LTSC.

- **Commercially Viable.** The "quality of implementation" can be varied to meet commercial needs. Implementers and vendors may create a spectrum of implementation qualities from which consumers may choose, such as varying levels of:
  - **Conformance:** from "strictly conforming" (maximum interoperability) to "conforming" (a variety of extended capabilities);
  - **System Performance:** a variety of performance metrics;
  - **Purchase and Maintenance Cost:** a variety of financial metrics;
  - **Interoperability.** The level of interoperability is related to the level of conformance to this Standard, related standards, and related specifications. However, conformance is subtly different from interoperability: conformance is the satisfaction, by an implementation (or a system), of the requirements of a standard or specification, while interoperability is the successful interaction among two or more implementations and the automation, to the level desired, of said interactions. For example, several combinations of interoperability are possible:
    - — Scenario #1: An implementation interoperates only among strictly conforming implementations. Example: The implementation might only include the features of this Standard, but no extensions or proprietary features are used.
    - — Scenario #2: An implementation interoperates with other implementations from the same vendor, user, or institution.
    - — Scenario #3: An implementation interoperates with a wide variety of user-specific, vendor-specific, institution-specific, and/or industry-specific extensions.

    Interoperability is dependent upon the kind of conformance to this Standard, and may be dependent upon conformance to features outside this Standard (e.g., extensions, such as related standards, specifications, and contractual agreements).
- **Extendable.** This Standard describes a set of features developed and agreed to in the formal consensus-building process. However, users, vendors, institutions, industries, and others will desire additional features (extensions) to support their specific needs. Conforming PAPI Learner implementations may use extensions, as permitted by implementations and as permitted data interchange interoperability requirements. As certain extensions become widely used, the consensus-building process may choose to incorporate these features into this Standard via a future Amendment or Revisions (see above). Thus, widely used extensions now may later become additional requirements that will be imposed upon future, strictly conforming (maximally interoperable) implementations. IEEE 1484.14.x "Extension Techniques" describes these standards lifecycle processes.

Note: PAPI Learner was initially developed for learning technology applications but may be easily extended to other types of human-related information such as medical and financial applications.

1.3 Normative wording vs. informative wording

Note: This subclause is informative and not normative.

This document contains two types of technical description:

- **Normative wording.** This wording places technical requirements on conforming implementations — normative wording is the essence of this Standard. Conformity assessment (e.g., conformance testing) is based solely on normative wording. Normative wording *excludes* introductory material, overview, rationale, footnotes, examples, bibliography, informative annexes, and sections labeled "*this section/clause/subclause is informative and not normative*".
- **Informative wording.** This wording is helpful, but not required, for understanding this document. Clause 1 and Annexes A-B and E are informative. Other informative wording is identified in individual cases. The Notes given provide clarification of the text, examples, and guidance — they do not contain technical requirements and do not form an integral part of this Standard.

1.4 Document organization (road map)

Note: This subclause is informative and not normative.

This Standard consists of 9 clauses and 5 annexes.

It is strongly recommended that the stakeholders Application Developer, Institutional Administrator, Security Administrator, Regulator (e.g., Legislator), and User (Learner) read Annex C, Stakeholder Perspectives, for additional guidance on reading this Standard.

The following is an overview of each section.

- **Clause 1 [Introduction]:** background information and a high-level summary of the features of this Standard.
- **Clause 2 [Normative References]:** normative wording that is incorporated by referring to other standards and specifications.
- **Clause 3 [Definitions]:** a list of terms and their definitions, and a list of abbreviations.
- **Clause 4 [Conformance]:** the technical requirements for claiming conformance to this Standard.
- **Clause 5 [Functionality]:** description of the purpose and use of PAPI Learner implementations.
- **Clause 6 [Conceptual Model]:** description of a "logical" (in contrast to "actual") implementation; the PAPI learner information types: personal, relations, security, preference performance, portfolio.
- **Clause 7 [Semantics]:** the meaning of data interchange across implementations and bindings.
- **Clause 8 [Bindings]:** the mapping of semantics to one or more codings, APIs, and protocols.

- **Clause 9 [Encodings]:** the representation and format of information as data formats, calling conventions, and communication layers.
- **Annex A [Bibliography, informative]:** references to related documentation.
- **Annex B [ISO/IEC 11404 Data Model Summary, informative]:** a summary of the data model in ISO/IEC 11404 notation.
- **Annex C [XML Coding Binding, conditionally normative]:** the XML coding binding of PAPI Learner data interchange.
- **Annex D [DNVP Coding Binding, conditionally normative]:** the RFC 822-Style (E-mail header) binding of PAPI Learner data interchange.
- **Annex E [Document Development, informative]:** a revision history and list of all outstanding issues with respect to this document.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this International Standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. IEEE and members of ISO and IEC maintain registers of currently valid Standards.

- ANSI X3.30:1998, "Representation of Calendar Date and Ordinal Date for Information Interchange"
- ANSI X3.42:1990, "Representation of Numeric Values in Character Strings for Information Interchange"
- ANSI X3.285:1998, "Metamodel for Data Representation"
- ANSI "American National Standard Dictionary for Information Technology (ANSDIT)"
- IEEE 1484.1, Learning Technology Systems Architecture.
- IEEE 1484.3, Learning Technology Glossary.
- IEEE 1484.12, Learning Object Metadata (LOM).
- IEEE 1484.13, Simple Human Identifiers.
- IEEE 1484.14, XML Bindings.
- IEEE 1484.15, Data and Control Transfer Protocol (DCTP).
- IEEE 1484.20, Competency Definitions.
- IETF RFC 822, Format of E-mail Messages and E-mail addresses (???)
- IETF RFC 2068, Hypertext Transfer Protocol (HTTP/1.1)
- ISO/IEC 2382 "Information Technology — Vocabulary" (multiple parts)
- ISO 8601, "Date and Time Formats"
- ISO/IEC 11404:1995, Language Independent Datatypes.
- ISO/IEC 11179, "Metadata Registries"
- ISO/IEC 15067-1, "Data and Control Transfer Protocol (DCTP)"
- ISO/IEC 15945 "Specification of Trusted Third Party Services to Support the Application of Digital Signatures"
- ISO/IEC 17799-1, "Code of Practice for Information Security Management"
- ISO/IEC 20944, "Metadata Access Service (MDAS)"
- W3C XML, Extensible Markup Language (XML) http://www.w3.org/TR/REC-xml

3 Definitions

Note: The following definitions are being/have been harmonized with the IEEE 1484.3 Glossary and Reference Materials.

3.1 Definitions incorporated via normative reference

Note: The following terms and their definitions have been incorporated via the normative references:

- ISO/IEC 2382 "Information Technology — Vocabulary" (multiple parts)
- ANSI "American National Standard Dictionary for Information Technology (ANSDIT)"
- IEEE 1484.3 Glossary and Reference Materials

3.2 accessor

A user, system, agent, or entity that attempts to access an asset within a security perimeter.

3.3 aggregate (datatype, value)

A generated datatype or value, each of whose datatypes or values is, in principle, made up of the component datatypes or values. The datatype or value is generated by applying an algorithmic procedure to combine the component datatypes or values. The component values are accessible via characterizing operations. The properties of the aggregate are independent of the properties of its components.

Example 1: An array aggregate contains components *all of the same type*. A characterizing operation uses an index (a number) to access individual components.

```
my_array:
    array (0..9) of (integer),  // an array of integers
my_array(4) // accessing element #4
```

Example 2: A record aggregate contains components, each component *individually typed and labeled*. A characterizing operation uses a element *name* (an *identifier* — a word) to access individual components.

```
A: record
(
    B: integer,
    C: void,
    D: characterstring(iso-10646-1),
),
A.B // accessing element labeled B
```

Note: This definition is adapted from ISO/IEC 11404.

3.4 application area

An industry or market segment for which a set of related applications are developed.

3.5 binding

An application or mapping from one framework or specification to another.

3.6 coding

(1) In information interchange, a formalized or structured representation of information. See Also: encoding.

(2) A process of representing information in some structure.

3.7 conditional data element

Within the appropriate context, an element of a data structure that is defined and required within an instance of the data structure, if certain conditions are satisfied.

The "conditional" nature of a data element is an obligation attribute.

See Also: extended data element; mandatory data element; obligation (data element); optional data element.

3.8 confidentiality

This subclause is informative and not normative.

A security technique that minimizes outbound security threats to an acceptable level by permitting retrieval or read access to authorized entities, and prohibiting retrieval and read access to unauthorized entities.

Note: Various security technologies may implement "confidentiality".

See Also: IEEE 1484.2.3 "PAPI Learner Information Security Notes", for more details on confidentiality-related issues.

3.9 consume (data)

To read data and then process it to the extent that some lexical or coding boundaries are discovered. A data consumer performs a limited number of translation phases.

Other Forms: consume data, data consumer, data consumption.

See Also: interpret (data); produce (data).

Note: Data is consumed before it is interpreted.

Example 1: In the following character stream:

```
<R>
  <A>123.45</A>
  <B>PQR</B>
  <C X="Y">Z</C>
</R>
<R>
```

```
        <D>JKL</D>
        <E>
          <F>XXX</F>
          <G>YYY</G>
        </E>
      </R>
```

a data consumer might recognize:

- there are two records, both with tags "**R**"
- the first "**R**" record contains three records with tags "**A**", "**B**", "**C**"
- the second "**R**" record contains two records with tags "**D**" and "**E**"

However, the data consumer:

- might not understand the meanings of tags: what does "**<B>...</B>**" mean?
- might not validate the tags: is "**<C>**" permitted to have the attribute "**X**"?
- might not validate the contents of the records: within record "**A**", is "**123.45**" a valid value?
- might limit the depth of its analysis: "**R**" is only explored one level deep to discover tags "**D**" and "**E**", but only a limited analysis (e.g., finding balanced tags) of the contents of "**E**" is performed such that tags "**F**" and "**G**" are not analyzed or discovered.

Thus, a data consumer might only have a partial understanding of an information structure.

Example 2: Below is an API example that distinguishes data consumption from data interpretation in a case where extended data of the implementation is indirectly used, yet the implementation is strictly conforming.

```
//// This example shows two files: the header "std_data.h",
//// and a strictly conforming application that includes
//// the header.

//////////////////////////////////////////////////////////////
//// The following is the include file "std_data.h"

struct std_data
{
        int std_element_1;    // Mandatory element.
        void *std_element_2;  // Optional element.
        int ext_element_3;    // Extended element.
};

//////////////////////////////////////////////////////////////
//// The strictly conforming application begins.

// Include the standard header (contents listed above)
#include "std_data.h"

struct std_data x;  // Declares "x" as standard data.

my_code()
{
        struct std_data y,z; // Declares "y" and "z".
```

```
                // Strictly conforming code, yet
                // extended element "ext_element_3"
                // is copied.
                memcpy(&y,&x,sizeof x);

                // Assign string to "std_element_2".
                // Assign length to "std_element_1".
                y.std_element_2 = "hello there";
                y.std_element_1 = strlen(y.std_element_2);

                // Still strictly conforming code, yet
                // extended element "ext_element_3"
                // is copied.
                memcpy(&z,&y,sizeof y);
        }
        ///////////////////////////////////////////////////////////////
```

This example is strictly conforming because the implementation only interprets or generates elements from a standard set, i.e., **`std_element_1`** and **`std_element_2`**. The **`memcpy`** (copy object in memory) operations are the equivalent *consume* and *produce* operations in this hypothetical API binding, while the direct element accesses (e.g., **`y.std_element_1`**) are the *interpret* and *generation* operations for this hypothetical API binding.

3.10 data application

An information technology application within an application area that is specified in the conformance clause or the conceptual model clause of a data interoperability standard.

Examples: data repository, data reader, data writer.

Note: A PAPI learner data application is from a PAPI Learner application. The latter is any information technology application that incorporates and conforms to PAPI Learner, while the former is a limited subset (i.e., data repository, data reader, data writer).

3.11 data instance

A data set rendered in some binding.

3.12 data object

A unit of data processing within the conceptual model of accessing implementations' data.

Note 1: A data object may be a data element or an implementation-defined object. Strictly conforming implementations only use or access data objects that are data elements.

Note 2: The behavior of a data object, further defined and constrained in the semantic definition, is a data structure. An instance of a data structure is a data set. A data set, further defined, constrained, and rendered in some binding is a data instance.

See Also: data element; data instance; data set; data structure.

3.13 data set

A data structure in its second definition, i.e., "an instance ... of data elements".

Note: A data set is independent of binding (binding-independent).

3.14 data structure

(1) The datatype of an aggregate of zero or more data elements.

(2) An instance of an aggregate of zero or more data elements.

Note 1: In a different context a data structure may be considered a whole, indivisible unit, i.e., in this context a data structure is a data element of some higher level data structure.

Note 2: The term "aggregate" is defined in ISO/IEC 11404.

Examples: a record; a set; a sequence; a list; an array.

3.15 distance (access, system)

Is constrained by bandwidth limitations or delays in communication systems such that applications are significantly affected.

3.16 distributed (access, system)

Uses the internet or wide area networks as the primary means of communication among subsystems and related systems.

3.17 encoding

The bit and byte format and representation of information.

3.18 extended data element

Within the appropriate context, an element of a data structure that is defined outside a standard, and may be used within an instance of the data structure, as permitted by data interchange participants and data interchange implementations.

The "extended" nature of a data element is an obligation attribute.

The "extended" nature of a data element is a conformance level feature (e.g., strictly conforming implementations vs. conforming implementations).

Example: mandatory extended data element, optional extended data element, conditional extended data element.

See Also: conditional data element; mandatory data element; obligation (data element); optional data element.

3.19 generate (data)

To transform data from its meaning to some form suitable for data interchange.

Example: To serialize a data structure according to a conceptual model without rendering the data in a specific coding or encoding.

See Also: interpret (data); produce (data).

3.20 group

A collection of users who share some common attributes.

Note 1: The nature and definition of the common attributes is outside the scope of this Standard.

Note 2: Groups may be created by administrators, users, or others.

Note 3: Depending on administrator policy, a user may belong to more than one group.

3.21 human identifier

A sign associated with a human.

See Also: learner identifier.

Note: Identifiers are specified by IEEE 1484.13 Simple Human Identifiers.

3.22 human information

Information that is primarily associated with, tracked by, tracked for, and about humans in information technology systems and learning technology systems.

3.23 implementation behavior

External observation, appearance, or action.

See Also: implementation-defined behavior; implementation value; undefined behavior; unspecified behavior.

3.24 implementation-defined behavior/value

Unspecified behavior or an unspecified value(s) where each implementation documents how the choice is made.

See Also: implementation behavior; undefined behavior/value; unspecified behavior/value.

Example: Permitting a maximum size, as measured in octets, of a coding.

3.25 implementation value

A quantifiable artifact associated with an implementation.

See Also: implementation behavior; implementation-defined behavior/value; undefined behavior/value; unspecified behavior/value.

3.26 inbound security threat

An external threat that breaches the security of a system and affects information inside the security perimeter.

Examples: The data injected into a communications stream; changing information; destroying information.

See Also: IEEE 1484.2.3 "PAPI Learner Information Security Notes", for related information.

3.27 information type

A category of information within a particular application area that is associated with some subset of application use and/or administration.

Example: "PAPI learner information" (an application area) contains six information types: (1) "PAPI learner personal information", (2) "PAPI learner relations information", (3) "PAPI learner security information", (4) "PAPI learner preference information", (5) "PAPI learner performance information", and (6) "PAPI learner portfolio information".

3.28 integrity (data)

This subclause is informative and not normative.

A technical policy about information security that reduces inbound security threats to an acceptable level.

Note: Data integrity may include: controlling the creation of information, controlling changes to information, or other techniques. The policy may be implemented by various security techniques, security technologies, security procedures, practices, etc..

See Also: IEEE 1484.2.3 "PAPI Learner Information Security Notes", for more details on data integrity.

3.29 interpret (data)

To process data to discover its meaning, to the extent required by this Standard.

Other Forms: interpret data, data interpreter, data interpretation.

See Also: generate (data); consume (data).

Note: Data is consumed before it is interpreted.

Example 1: In the following character stream:

```
<R>
  <A>123.45</A>
  <B>PQR</B>
```

```
    <C X="Y">Z</C>
  </R>
  <R>
    <D>JKL</D>
    <E>
      <F>XXX</F>
      <G>YYY</G>
    </E>
  </R>
```

a data consumer might recognize:

- there are two records, both with tags "**R**"
- the first "**R**" record contains three records with tags "**A**", "**B**", "**C**"
- the second "**R**" record contains two records with tags "**D**" and "**E**"

Because only these tags are recognized, only these tags are candidates for data interpretation. Assuming tag **"E"** represents an extended data element, a data interpreter might only recognize the standardized tags **"A"**, **"B"**, **"C"**, and **"D"**.

Based on (1) the separation of the "consume" and "interpret" phases of translation, and (2) a particular standards binding (XML-like in this case), an application might only interpret the standardized features A, B, C, and D.

As described above, an application that combines data consumption and data interpretation, but only interprets standardized data elements, might be strictly conforming data reader.

3.30 learner

An individual engaged in acquiring knowledge or skills with a learning technology system. See Also: learner entity.

3.31 learner entity

An entity that represents a collective learner, such as team learning.

Note: The IEEE 1484.1 LTSA distinguishes between "learner" and "learner entity". For PAPI learner information, there is no distinction between "learner" and "learner entity" — both are referred to as "learner".

3.32 learner identifier

A sign associated with a learner.

Note 1: A learner may have more than one learner identifier — a non-singular identifier. The policy of singularity or non-singularity of identifiers is outside the scope of this Standard.

Note 2: The term "moniker" is believed to come from Shelta via thieves' slang. "Shelta" is defined as: "an esoteric jargon based in part on the systematic alteration of Irish and Gaelic and still spoken in some parts of England and Ireland by tinkers, vagrants, etc.".

Examples: IEEE 1484.13 Simple Human Identifiers, passport numbers, E-mail addresses, social security numbers.

3.33 learner information

The intersection of general learning technology information and human information for learners or learner entities.

3.34 learner performance granularity

The relative size, scope, or detail of a learner performance information.

Note: Learner performance records may have different granularities.

Example: recording key clicks (e.g., seconds of learning); minute-by-minute data samples of learner progress (e.g., minutes of learning); lesson scores (e.g., days of learning); semester grades (e.g., months of learning); professional certifications (e.g., years of learning).

3.35 learner profile

This subclause is informative and not normative.

Information about a learner used by specific learning technology components, learning technology applications, and learner administration. A subset of learner information, in general.

Example 1: A learner profile includes information such as personal, relations, security, preference, performance, portfolio, and, possibly, other types of information.

Example 2: PAPI learner information is an example of a learner profile.

3.36 learning management system

A system that (1) schedules learning resources; (2) assists, controls, and/or guides the learning process; and (3) analyzes and reports learner performance.

3.37 locale-specific behavior

Behavior that depends on local conventions of nationality, culture, language, institution, etc., which is documented by each implementation.

3.38 longevity (data element)

An attribute of a data element specification that indicates intention for incorporation into past, present, or future editions of a standard.

See Also: obligation (data element); obsolete data element; reserved data element.

Note: Longevity attributes are independent of obligation attributes.

Example 1: An obsolete data element might have been intended for inclusion in past editions of this Standard, but is intended to be excluded in future editions of this Standard.

Example 2: A reserved data element might not have been included in past editions of this Standard, and might be intended for inclusion in future editions of this Standard.

3.39 mandatory data element

Within the appropriate context, an element of a data structure that is defined and required within an instance of the data structure.

The "mandatory" nature of a data element is an obligation attribute.

See Also: conditional data element; extended data element; obligation (data element); optional data element.

3.40 nomadic (access, system)

(1) The appearance of continuity of service across separate communication sessions and geographic locations.

(2) Sometimes-disconnected from the networks used for communication among its subsystems and related systems.

Note: Also known as "sometimes-connectivity" and/or "sometimes-roaming".

3.41 obligation (data element)

The requirements and permissibility of data elements that determine the validity of a data structure.

See Also: longevity (data element); conditional data element; extended data element; mandatory data element; optional data element.

Note: Obligation attributes are independent of longevity attributes.

Example: A data structure **`X`**, has four elements: **`A`** and **`B`** are mandatory, **`C`** is optional, and **`D`** is conditional if **`B`** has the value **`true`**. The following are sample valid and invalid data structures:

```
( A=123 )                 // invalid: missing mandatory element B
( A=123, B=false )        // valid
( A=123, B=true )         // invalid: missing conditional element D
( A=123, B=true, D=17 )   // valid
( A=123, B=false, D=17 )  // valid: allowable because the example
                          // "conditional" wording above only
                          // makes requirements and makes no
                          // prohibitions
( A=123, B=nil, C=345 )   // valid
```

3.42 obsolete data element

Within the appropriate context, an element of a data structure that is defined but should not be used within an instance of the data structure.

The "obsolete" nature of a data element is a longevity attribute.

See Also: longevity (data element); reserved data element.

Note: The use of obsolete data elements is deprecated and their specification may be removed from future revisions of a standard.

3.43 optional data element

Within the appropriate context, an element of a data structure that is defined and permitted, but not required within an instance of the data structure.

The "optional" nature of a data element is an obligation attribute.

See Also: conditional data element; extended data element; mandatory data element; obligation (data element).

3.44 outbound security threat

The theft or unauthorized duplication of information such that the information becomes available outside the security perimeter or no longer remains inside the security perimeter.

Examples: Snooping network packets; taking information.

See Also: IEEE 1484.2.3 "PAPI Learner Information Security Notes", for related information.

3.45 PAPI Learner

An abbreviated name for this Standard.

Note: The term "PAPI Learner" (with an upper case "L") is used when the sentence could be rewritten by replacing "PAPI Learner" with "IEEE 1484.2", yet the sentence retains its meaning.

Examples: When referring to this Standard or its preliminary documents, the following constructions are appropriate: "PAPI Learner Standard" and "PAPI Learner, draft *X*" (both refer to drafts of this document); "PAPI Learner implementation" (an implementation of this Standard); and "PAPI Learner data interchange" (data interchange facilitated by this Standard).

3.46 PAPI learner

A subset of public and private information (human information) about learners as defined by this Standard.

Note 1: The term "profile information" is a generic name and this Standard is *one* description of this "profile information".

Note 2: The term "PAPI learner" (with a lower case "L") is the term defined in this Clause (Definitions), is used as a prefix of a term in this Clause, or is used when a sentence does not retain its meaning by replacing "PAPI learner" with "IEEE 1484.2".

Example 1: "PAPI learner API" is already defined in this Clause, thus "PAPI *Learner* API" is an appropriate term.

Example 2: "PAPI learner application" has a meaning that is different than simply connecting the "IEEE 1484.2" adjective with the term "application". The term "PAPI *learner* application" has a definition that is different than just all "applications" of "IEEE 1484.2" (i.e., the term "PAPI *Learner* applications"). Typically, the context of a term's usage avoids these kind of ambiguities.

3.47 PAPI learner application

An information technology application that uses codings, APIs, and/or protocols that conform to this Standard.

3.48 PAPI learner data application

A data repository, data reader, or data writer, as specified in this Standard.

3.49 PAPI learner extensions

Extended data elements and/or data services extensions.

Note: Strictly conforming implementations do not use extensions. Conforming implementations may use extensions.

3.50 PAPI learner information

See "PAPI learner".

3.51 PAPI learner record

A collection of learner information represented as defined by this Standard.

3.52 PAPI learner record reference

An identifier that points to a PAPI learner record.

3.53 PAPI learner system

See: PAPI learner application.

3.54 PAPI server

See: PAPI learner server.

3.55 privacy (data)

This subclause is informative and not normative.

A technical policy about information security that reduces outbound security threats to an acceptable level. Privacy may include: controlling the copying of information, controlling transfer of information, or other techniques. The policy may be implemented by various security techniques, security technologies, security procedures, practices, etc..

See Also: IEEE 1484.2.3 "PAPI Learner Information Security Notes", for more details on data privacy.

3.56 produce (data)

To process data to the extent that lexical or coding boundaries are defined and then write the resultant data.

Other Forms: produce data, data producer, data production.

See Also: generate (data); consume (data).

Note: Data is generated before it is produced.

3.57 public and private information

Human-related information, of varying degrees of public and private accessibility, exchanged and administered within information technology systems and communication networks.

3.58 repository

A collection of data sets and data access methods for storing, indexing, searching, and retrieving information.

3.59 reserved data element

Within the appropriate context, an element of a data structure that is not defined and not permitted to be used within an instance of the data structure.

The "reserved" nature of a data element is a longevity attribute.

See Also: longevity (data element); obsolete data element.

3.60 role-based access control

A security technique for authentication that authorizes operations or allows access to resources based upon the user's identity and his/her relationship to other users.

Example 1: A teacher has read/write access to the grades for his/her students (role: "the teacher of the student"), but no access to other students' grades.

Example 2: A principal has read-only access to the grades of all of his/her teachers' students (role: "the principal of the teachers of the students"), but the principal is not permitted to change any grades.

3.61 security administrator

A person who is responsible for security management within a security perimeter.

Note: A security administrator may be concerned about several types of security, including information security, physical security, financial security, and personal security.

3.62 security information

This subclause is informative and not normative.

Information that supports a security policy.

Note 1: The term "security information" has a variety of meanings in various contexts.

Note 2: This Standard only describes learner security information, and only describes security information to the extent that a variety of security implementations are *possible*. This Standard makes no requirements for specific security policies, procedures, techniques, or technologies.

3.63 security perimeter

A continuous, closed partition that separates the "inside" from the "outside". The "inside" of the security perimeter is intended to be secure. The "outside" of the security perimeter may not be secure.

3.64 security perimeter integrity

A level of security protection such that inbound security threats and outbound security threats are maintained at an acceptable level of risk.

Example 1: The loss of security perimeter integrity implies that (1) the level of security protection is less than an acceptable level; (2) the inbound security threats have increased to more than an acceptable level; or (3) the outbound security threats have increased to more than an acceptable level.

Example 2: User permissions and administration are aspects of security perimeter integrity.

See Also: IEEE 1484.2.3 "PAPI Learner Information Security Notes", for related information.

3.65 security strength

The degree of security that characterizes the implementation of a security perimeter.

Note: The following features may characterize security strength:

- Quality Level: What level of security is provided? Examples of security levels: minimal security, auditing-capable, provable design.
- Contingencies: What happens when the system is violated? What fallbacks are available? What is the extent of the damage?
- Penalties: What happens to violators of the security mechanism?

Example: Security strength might concern the number of bits used in encryption keys.

3.66 simple human identifier

A human identifier specified by IEEE 1484.13 Simple Human Identifiers.

See also: human identifier.

3.67 smallest permitted maximum

For implementation-defined values, the smallest permitted maximum value.

Example: "The smallest permitted maximum string length of element X shall be 17."

3.68 undefined behavior/value

Implementation behavior or an implementation value(s) for which a standard imposes no requirements.

See Also: implementation behavior; implementation value; implementation-defined behavior/value; unspecified behavior/value.

Example 1: Possible undefined behaviors include, but are not limited to:

- ignoring the situation completely
- unpredictable results
- behaving in a documented manner characteristic of the environment
- terminating processing

Example 2: Possible undefined values include infinities, null values, and "Not A Number".

3.69 unspecified behavior/value

Implementation behavior or an implementation value(s) for which a standard provides two or more possibilities and imposes no further requirements on which possibility is chosen in any instance.

See Also: implementation behavior; implementation value; implementation-defined behavior/value; undefined behavior/value.

Example 1: An application's choice of algorithm for creating object identifiers.

Example 2: The order in which procedure call parameters are pushed on a calling stack.

3.70 user

A human, his/her agent, or a surrogate that interacts with information technology systems.

Note: For PAPI Learner, a user is typically concerned about the proper creation, correction, or destruction of records, the potential use of records; the security of records, and the usefulness of records within information technology systems. Users may be learners, teachers, employers, administrators, etc..

3.71 Acronyms and abbreviations

- ANSI: American National Standards Institute
- API: Application Programming Interface
- ASN.1: Abstract Syntax Notation One; also known as ISO/IEC 8824 and 8825
- DCTP: Data and Control Transfer Protocol
- HTTP: Hypertext Transfer Protocol

- I18N: internationalization; the internationalized spelling of "internationalization" (the letter "I", followed by 18 letters, then the letter "N")
- ICS: Implementation Conformance Statement
- IEC: International Electrotechnical Commission
- IEEE: Institute of Electrical and Electronic Engineers
- IETF: Internet Engineering Task Force
- ISO: International Organization for Standardization
- L10N: localization; the internationalized spelling of "localization" (the letter "L", followed by 10 letters, then the letter "N")
- LID: Language Independent Datatypes; also known as ISO/IEC 11404
- LMS: learning management system
- LTSA: Learning Technology Systems Architecture
- LTSC: Learning Technology Standards Committee
- MDAS API: Metadata Access Service API
- PAPI: Public and Private Information specification
- RFC: Request for Comments; a citation prefix for specifications developed by the Internet Engineering Task Force
- SPM: smallest permitted maximum
- W3C: World Wide Web Consortium
- XML: Extensible Markup Language

4 Conformance

In this Standard, "shall" is to be interpreted as a requirement on an implementation; "shall not" is to be interpreted as a prohibition. If a "shall" requirement or "shall not" prohibition is violated, the behavior is undefined. Undefined behavior is otherwise indicated in this Standard by the words "undefined behavior" or by the omission of any explicit definition of behavior. There is no difference in emphasis among these three; they all describe "behavior that is undefined".

Note: Implementations claim conformance to particular features of this Standard in their implementation conformance statement.

Rationale

The first sentence in the above paragraph might appear to be unnecessary because the meaning and usage of "shall" is well understood in standards, but once the remaining sentences are added, the first sentence becomes necessary.

Conformance partly concerns the structure of data sets and partly concerns the behavior of systems. In other words, conformance has both a non-behavioral and a behavioral dimension, and both are important. The term "behavior" is used in a general sense, e.g., a data set doesn't exhibit "behavior", but implementations that store, retrieve, generate, and interpret data sets all exhibit "behavior" — the notions of "undefined", "implementation-defined", and "unspecified" behaviors are defined, even in the context of a data set.

4.1 Conformance level

The following subclauses define strictly conforming implementations and conforming implementations. In the context of conformance, the terms "support", "use", "test", "access", and "probe" are defined in subclause 4.3, Coding Conformance, subclause 4.4 API Conformance, subclause 4.5, Protocol Conformance, and subclause 4.6, Data Application Conformance.

Rationale

The distinction between "strictly conforming" and "conforming" implementations is necessary to address the simultaneous needs for interoperability and extensions. This Standard describes specifications that promote interoperability. Extensions are motivated by needs of users, vendors, institutions, and industries (1) that are not directly specified by this Standard, (2) that are specified and agreed to outside this Standard, and (3) that may serve as trial usage for future editions of this Standard.

4.1.1 Subsets

An implementation shall support at least one PAPI learner information type (personal, relations, security, preference, performance, portfolio). Implementations that do not support all PAPI learner information types shall indicate which subset is supported in the implementation conformance statement and in the conformance label(s). See subclause 4.2, Conformance Labels.

Note: Implementations should use automated techniques to convey subset information so that interoperability problems can be avoided.

4.1.2 Strictly conforming implementations

A strictly conforming implementation shall be at least one of: a strictly conforming coding, a strictly conforming API, a strictly conforming protocol, or a strictly conforming data application.

A strictly conforming implementation:

1. shall support all mandatory and optional data elements;
2. shall not use, test, access, or probe for any extension features or extended data elements;
3. shall not exceed limits or smallest permitted maximum values specified by this Standard; and
4. shall not interpret or generate data elements that are dependent on any unspecified, undefined, implementation-defined, or locale-specific behavior.

Note: The use of extension features or extended data elements is undefined behavior.

4.1.3 Conforming implementations

A conforming implementation shall be at least one of: a conforming coding, a conforming API, a conforming protocol, or a conforming data application.

A conforming implementation:

1. shall support all mandatory and optional data elements;
2. may use, test, access, or probe for extension features or extended data elements, as permitted by the implementation and data interchange participants, as long as the meaning and behavior of strictly conforming implementations is unchanged;
3. shall not support or use extension features or extended data elements that change the meaning or behavior of strictly conforming implementations;
4. may exceed limits or smallest permitted maximum values specified by this Standard, and to the extent permitted by the implementation; and
5. may interpret or generate data elements that are dependent on implementation-defined, locale-specific, or unspecified behavior.

Note 1: The use of extension features or extended data elements is undefined behavior.

Note 2: All strictly conforming implementations are also conforming implementations.

Note 3: An implementation does not conform to this Standard if it redefines Standard features via extension methods, and these features change the meaning or behavior of strictly conforming implementations.

4.1.4 Non-conforming implementations

This subclause is informative and not normative.

An implementation that does not conform to this Standard (either strictly conforming or merely conforming), is a non-conforming implementation.

4.1.5 Obligation of data elements

This subclause is informative and not normative.

There are four types of obligation attributes for data elements: mandatory, optional, conditional, and extended. The obligation attribute concerns the validity of the data structure.

4.1.5.1 Mandatory data elements

Mandatory data elements are always required for the data structure to be valid. All data sets (and data instances) are required to include these elements. All data applications are required to support these elements.

An implementation that does not support or include one or more mandatory data elements is a non-conforming implementation.

4.1.5.2 Optional data elements

Optional data elements are permitted, but not required, for the data structure to be valid. A data set (and data instance) is permitted, but not required, to include these data elements. Because all data repositories and data readers are required to support all valid data sets, effectively, data repositories and data readers are required to support all optional data elements. This might be confusing because "optional" is *not* optional for data repositories and data readers — the obligation attribute "optional" applies to the *validity of the data structure* ("optional" is optional for instances of the data structure). A data writer is required to generate and produce the optional elements of each data instance that is generated and produced.

An implementation that does not support one or more optional data elements is a non-conforming implementation.

If an implementation includes or supports these data elements, their use is specified by this Standard.

An implementation that includes or supports an optional data element, but includes or supports it in ways that are inconsistent with this Standard, is a non-conforming implementation. The attribute "optional" does *not* imply that the implementation has license to implement the data element in any way ("at the option of the implementor"); if the data element is implemented, its requirements are specified in this Standard.

4.1.5.3 Conditional data elements

Conditional data elements are required, but their requirement is dependent upon certain conditions (as defined elsewhere in this Standard). Each conditional data element may individually have a set of conditions. If the conditions are met, the data element is required to be included for the data structure to be valid. Thus, a data set (and data instance) is required to include these elements if, individually, each condition is met. By the same reasoning as for optional data elements (above), all data repositories and data readers are required to support all conditional data elements. By the same reasoning above, a data writer is required to support all the conditional elements for each and every data instance generated and produced.

An implementation that does not support one or more conditional data elements is a non-conforming implementation.

An implementation that includes or supports a conditional data element, but includes or supports it in ways that are inconsistent with this Standard, is a non-conforming implementation.

4.1.5.4 Extended data elements

Extended data elements are not permitted within strictly conforming implementations.

Extended data elements are permitted within conforming implementations to the extent that the implementation individually supports each extended data element, i.e., (1) the implementation allows and uses specified extended data elements, (2) the data interchange participants allow and use specific data elements, and (3) other extended data elements are not used.

For conforming implementations that support extended elements, these elements individually may have their own obligation attributes, e.g., it is possible to have mandatory extended data elements, optional extended data elements, and conditional extended data elements. These obligation attributes determine the validity of the data structure in the context of extended data elements, e.g., an optional extended data element (1) permits but does not require the data element for the data structure to be valid, (2) for conforming implementations that support this extended data element.

Note: Mandatory extended data elements can cause interoperability problems because a mandatory extended data element (1) requires the data element to exist for the data structure to be valid, (2) for conforming implementations that support this extended data element. In other words, (1) only implementations that support this extended data element are interoperable; and (2) *no* strictly conforming implementations will interoperate because *extended features are required* for interoperability.

There are no generic techniques or methods for both supporting extended data elements or extension features and supporting full semantic interoperability; there are only specific techniques and methods for supporting extended data elements (e.g., supported to the extent allowed, as above).

The use of extended data elements outside these circumstances (unsupported environments) causes undefined behavior, which might be:

- appropriate, e.g., ignoring an offending data element if it is an unimportant feature

- inappropriate, e.g., ignoring an offending data element if it is an important feature, such as a security classification
- innocuous, e.g., error messages
- disruptive, e.g., error messages
- predictable, e.g., a program aborting, exiting ungracefully, exiting unexpectedly, or "hanging" indefinitely
- unpredictable, e.g., a program aborting, exiting ungracefully, exiting unexpectedly, or "hanging" indefinitely

There is no correct generalized method for handling undefined behavior. Any particular method for handling undefined behavior can be desirable, undesirable, or both.

Some bindings "relax" the processing of unrecognized extended data elements. Normally, extended data elements create *undefined behavior* but certain bindings "relax" these requirements to *implementation-defined behavior* or even *ignoring* unrecognized extended data elements — both of these "relaxed" processing requirements (implementation-defined behavior; ignoring unrecognized or extended data elements) can be less disruptive. IEEE 1484.14.x "Extensions Techniques", contains informative wording on the relationship among data modeling, conformance levels, interoperability, industry concerns, and the standards process.

Extended data elements are both an obligation (data modeling) feature and a conformance level feature (strictly conforming vs. conforming).

4.1.6 Longevity of data elements

This subclause is informative and not normative.

The following longevity attributes indicate intentions for incorporation into past, present, or future editions of this Standard.

Longevity attributes are independent of obligation attributes.

4.1.6.1 Obsolete data elements

Obsolete data elements are defined in the current edition of this Standard and may be defined in prior editions. The "obsolete" feature indicates that the definition of the data element is intended to be removed from future editions of this Standard.

Implementations should not use obsolete data elements. Implementations that do use obsolete data elements should plan accordingly for future editions of this Standard.

An implementation's use of an obsolete data element does not imply that the implementation is non-conforming. Strictly conforming implementations and conforming implementations may still use obsolete data elements for <u>this</u> edition of the Standard.

The "obsolete" feature is independent of the obligation attribute, so there might be obsolete mandatory data elements, obsolete optional data elements, obsolete conditional data elements, and obsolete extended data elements.

4.1.6.2 Reserved data elements

Reserved data elements are not defined in this edition of the Standard. Data elements may be reserved because (1) they were defined in previous edition(s) of this Standard, or (2) they will be defined in some future edition(s) of this Standard.

A reserved data element is not permitted in a strictly conforming implementation.

A "reserved data element" might be used in a conforming implementation if (1) the reserved data element were defined, (2) it were defined as an extended data element, and (3) the extended data element were "supported" by implementations and data interchange participants (see below). In other words, a particular implementation extends or overrides (the non-definition of) the "reserved data element" by defining implementation extensions.

Although the "reserved" feature is independent of the obligation attribute, a reserved data element has no definition. Therefore, there are no reserved mandatory data elements, no reserved optional data elements, no reserved conditional data elements, and no reserved extended data elements because mandatory data elements, optional data elements, conditional data elements, and extended data elements all imply a definition of a data element, which conflicts with the undefined nature of "reserved".

Data elements that are defined, but are to be incorporated into future editions of this Standard, are extended data elements (i.e., they are *not* reserved data elements). As these extended data elements become incorporated into a future edition, they will become mandatory data elements, optional data elements, or conditional data elements.

Extended data elements may be defined (1) in an informative Annex in this Standard, (2) in a conditionally normative Annex in this Standard, or (3) in a specification outside this Standard.

Extended data elements are not required for this edition of the Standard, i.e., (1) extended data elements are prohibited for strictly conforming systems; (2) extended data elements are not required for conforming systems; and (3) extended data elements, if defined, are not in the Clauses of this Standard.

Some bindings "relax" the processing of unrecognized data elements, such as reserved data elements. Normally, reserved data elements create *undefined behavior* but certain bindings "relax" these requirements to *implementation-defined behavior* or even *ignoring* unrecognized or reserved data elements — both of these "relaxed" processing requirements (implementation-defined behavior; ignoring unrecognized or reserved data elements) can be less disruptive. IEEE 1484.14.x "Extensions Techniques", contains informative wording on the relationship among data modeling, conformance levels, interoperability, industry concerns, and the standards process.

Conforming implementations may use extended data elements to the extent permitted by the implementation and data interchange participants. See subclause 4.1.4.4, Extended Data Elements, above, for further details.

Reserving a data element so that it cannot be overridden by extended data elements is achieved by defining an optional data element with ISO/IEC 11404 datatype **`void`**.

4.1.7 Recursive/contextual nature of obligation/longevity

This subclause is informative and not normative.

An obligation attribute or a longevity attribute of an aggregate data element applies to the aggregate itself, but only indirectly to its components. In the context of the existence of an aggregate and its components, each component individually has its own obligation and longevity attributes (among other attributes). This determination of context and obligation/longevity attributes is applied recursively for all aggregate data elements.

Example: A data element **`X`** is optional, and **`X`** has two subelements: **`Y`** is mandatory and **`Z`** is optional. Letting the notation **`P.Q`** represent the subelement **`Q`** of **`P`**, then

- if **`X`** does not exist, then **`X.Y`** and **`X.Z`** cannot exist; stated differently, if **`X.Y`** or **`X.Z`** exists, then **`X`** exists
- if **`X`** exists, then **`X.Y`** is required to exist for all conforming implementations
- if **`X`** exists, then **`X.Z`** is permitted to exist for all conforming implementations
- if **`X`** exists and **`X.Y`** does not exist, then the implementation is non-conforming

Thus, **`Y`** only becomes mandatory if **`X`** exists.

4.2 Conformance labels

A conformance labels may summarize implementation conformance statements (ICSs). Conformance labels should be used to convey ICS information via manual, semi-automated, and automated methods. The methods and techniques for associating or affixing a conforming label are outside the scope of this Standard.

If an implementation does not support all information types, the conformance label shall indicate which subset is supported using the notation "Subset *one-letter-subset-list*", where the one abbreviations are defined in subclause 6.1, Learner Information. Example: An implementation that only supports PAPI learner preference information (M) and PAPI learner performance information (G) would indicate this subset with "Subset MG".

For bindings descriptions in conformance labels, bindings shall be listed in the order of codings, APIs, and protocols. For the codings, APIs, and protocols listed, each coding binding shall operate with each API binding, each API binding shall operate with each protocol binding, and each coding binding shall operate with each protocol binding. Note: If all combinations of the listed codings, APIs, and protocols do not interoperate, then multiple conformance labels may be used to indicate claims of conformance. Example 1: "XML and DNVP coding/ Java and JavaScript API" implies that the implementation conformance to all four combinations: XML-Java, DNVP-Java, XML-JavaScript, DNVP-JavaScript. Example 2: In the previous example, if the implementation did not support the XML-JavaScript combination, then two conformance labels "XML and DNVP coding/ Java API" and "DNVP coding/ JavaScript API" might describe implementation conformance claim.

The following is a summary of the possible implementation varieties used in implementation conformance statements (ICS) and their conformance labels:

- **Strictly Conforming PAPI Leaner *[Subset]* Data Set:** binding-independent; all mandatory data elements shall exist; some optional data elements may exist; extended data elements shall not exist.
- **Conforming PAPI Learner *[Subset]* Data Set:** binding-independent; all mandatory data elements shall exist; some optional data elements may exist; some extended data elements may exist.
- **Strictly Conforming PAPI Learner *[Subset]* *binding* Data Instance:** a binding shall be specified; all mandatory data elements shall exist; some optional data elements may exist; extended data elements shall not exist. Example: The file **`"papi.xml"`** is a Strictly Conforming PAPI Learner XML Data Instance.
- **Conforming PAPI Learner *[Subset]* *binding* Data Instance:** a binding shall be specified; all mandatory data elements shall exist; some optional data elements may exist; some extended data elements may exist. Example: The file **`"papi.txt"`** is a Conforming PAPI Leaner DNVP Data Instance.
- **Strictly Conforming PAPI Learner *[Subset]* *binding(s)* Data Repository:** binding(s) shall be specified; shall support storing/retrieving all mandatory data element attributes, shall support storing/retrieving all optional data elements; data interchange applications shall not attempt to store/retrieve extended data elements. Example: The server XYZ is a Strictly Conforming PAPI Learner XML-coding/ SOAP-protocol Data Repository.
- **Conforming PAPI Learner *[Subset]* *binding(s)* Data Repository:** binding(s) shall be specified; shall support storing/retrieving all mandatory data elements, shall support storing/retrieving all optional data elements; may support storing/retrieving some extended data elements. The server XYZ is a Conforming PAPI Learner DNVP-coding/ DCTP-protocol Data Repository.
- **Strictly Conforming PAPI Learner *[Subset]* *binding(s)* Data Reader:** binding(s) shall be specified; only mandatory and optional data elements are interpreted, but no extended data elements are interpreted. Example: The import tool XYZ is a Strictly Conforming PAPI Learner XML-coding/ Java-API Data Reader.
- **Conforming PAPI Learner *[Subset]* *binding(s)* Data Reader:** binding(s) shall be specified; mandatory and optional data elements are interpreted and some extended data elements may be interpreted. Example: The import tool XYZ is a Conforming PAPI Learner DNVP-coding/ HTTP tunneling-protocol Data Reader.
- **Strictly Conforming PAPI Learner *[Subset]* *binding(s)* Data Writer:** binding(s) shall be specified; shall generate all mandatory data elements; may generate optional data elements; shall not generate extended data elements. Example: The export tool XYZ is a Strictly Conforming PAPI Learner DNVP-coding/ JavaScript-API Data Writer.
- **Conforming PAPI Learner *[Subset]* *binding(s)* Data Writer:** binding(s) shall be specified; shall generate all mandatory data elements; may generate optional data elements; may generate extended data elements. Example: The export tool XYZ is a Conforming PAPI Learner XML-coding/ SOAP-protocol Data Writer.
- **Strictly Conforming PAPI Learner *[Subset]* *binding(s)* API Environment:** binding(s) shall be specified; shall support all mandatory and optional data elements; extended data elements and extended services shall not be probed by applications of the API binding. Example: The software development kit XYZ is a Strictly Conforming PAPI Learner Java API environment.

- **Conforming PAPI Learner *[Subset]* *binding(s)* API Environment:** binding(s) shall be specified; shall support all mandatory and optional data elements. Example: The software development kit XYZ is a Conforming PAPI Learner JavaScript API environment.
- **Strictly Conforming PAPI Learner *[Subset]* *binding(s)* API Application:** binding(s) shall be specified (in order: codings, APIs, protocols); shall support all mandatory and optional data elements; extended data elements and extended services shall not be probed. Example: The application XYZ is a Strictly Conforming PAPI Learner C++-API Application.
- **Conforming PAPI Learner *[Subset]* *binding(s)* API Application:** binding(s) shall be specified; shall support all mandatory and optional data elements; extended data elements and extended services may be used to the extent permitted by data interchange participants and to the extent permitted by specifications external to this Standard. Example: The application XYZ is a Conforming PAPI Learner Perl Application.
- **Strictly Conforming PAPI Learner *[Subset]* *binding(s)* Protocol:** binding(s) shall be specified; shall support all mandatory and optional data elements; extended data elements and extended services shall not be probed by applications of the protocol binding. Example: The back office gateway XYZ is a Strictly Conforming PAPI Learner XML-coding/ SOAP Protocol.
- **Conforming PAPI Learner *[Subset]* *binding(s)* Protocol:** binding(s) shall be specified; shall support all mandatory and optional data elements; extended data elements and extended services may be used to the extent permitted by data interchange participants and to the extent permitted by specifications external to this Standard. Example: The back office gateway XYZ is a Conforming PAPI Learner DNVP-coding/ C++-API/ DCTP protocol.

Note: An implementation may claim more than one type of conformance in its implementation conformance statement (ICS).

4.3 Coding conformance

A strictly conforming PAPI learner coding shall be at least one of: a strictly conforming data set, or a strictly conforming data instance.

A conforming PAPI learner coding shall be at least one of: a conforming data set, or a conforming data instance.

A PAPI learner coding shall conform to Clause 5, Functionality; conform to Clause 6 Conceptual Model; and, conform to the datatypes specified in Clause 7, Semantics.

4.3.1 Data set conformance

Data set conformance is independent of binding.

A strictly conforming data set shall be a set of data that: (1) is structured independent of binding, (2) strictly conforms to the functionality, conceptual model, and semantics of this Standard, (3) shall include all mandatory data elements, (4) may include optional data elements, and (5) shall not include extended data elements.

A conforming data set shall be a set of data that: (1) is structured independent of binding, (2) conforms to the functionality, conceptual model, and semantics of this Standard (3) shall include all mandatory data elements, (4) may include optional data elements, and (5) may include extended data elements.

Conformity assessment of data sets shall be performed by (1) rendering the data set in ISO/IEC 11404 notation, and (2) verifying the requirements described by this Standard.

4.3.2 Data instance conformance

A strictly conforming data instance shall (1) be a strictly conforming data set, and (2) strictly conform to at least one PAPI learner coding.

A conforming data instance shall (1) be a conforming data set, and (2) conform to at least one PAPI learner coding.

Note 1: The term "PAPI learner coding" is used in the two paragraphs above and its requirements are specified in the third paragraph of subclause 4.3, Coding Conformance.

Note 2: The difference between a strictly conforming/conforming data set, a strictly conforming/conforming coding, and a strictly conforming/conforming data instance is: (1) a data set is an instance of data that is independent of binding, (2) a coding can refer to an instance of data, a set of instances of data, or a syntax of instances of data, and (3) a strictly conforming/conforming data instance is associated with a specific binding.

Definitions: support, use

In the context of conformance, the terms "support" and "use" are defined individually in each PAPI leaner coding binding.

Definitions: test, access, probe

In the context of conformance, the terms "test", "access", and "probe" are defined as the null operation, i.e., for data instance conformance, the operations "test", "access", and "probe" perform no operations and have no effect.

Rationale

In addition to the three application conformance perspectives (data repository, data reader, data writer), there is a fourth perspective on conformance: the data instance. Users will want to claim conformance for particular data instances ("My PAPI learner information conforms to the Standard").

4.4 API conformance

A strictly conforming PAPI learner API shall strictly conform to at least one PAPI learner API binding.

A conforming PAPI learner API shall conform to at least one PAPI learner API binding.

A PAPI learner API shall conform to Clause 5, Functionality; conform to Clause 6 Conceptual Model; and, conform to the operations and datatypes specified in Clause 7.

Definitions: support, use, test, access, probe

In reference to PAPI Learner conformance, the following terms are defined in the context of API conformance: a "supported" feature is one that may be used by any application of the PAPI learner API; a feature is "used" if it is read, written, or operated upon by an application of the PAPI learner API; a feature is "tested" if an application of the PAPI learner API inquires about the existence of said feature; a feature is "accessed" if an application of PAPI learner API attempts to read or write data associated with the feature; a feature is "probed" if an application implicitly tests the existence of a feature by attempting to *use* the feature (see "use" above) within a "safe" environment that does not cause undefined behavior.

Note: API conformance makes requirements on both the API binding and on applications that use the API binding, i.e., conformity assessment of a PAPI implementation based on API conformance is determined by proper definition of the API and proper use of the API.

4.5 Protocol conformance

A strictly conforming PAPI learner protocol shall strictly conform to at least one PAPI learner protocol binding.

A conforming PAPI learner protocol shall conform to at least one PAPI learner protocol binding.

A PAPI learner protocol shall conform to Clause 5, Functionality; conform to Clause 6 Conceptual Model; conform to the operations and datatypes specified in Clause 7, Semantics; and define methods for setup and knockdown of supporting communication networks.

Definitions: support, use, test, access, probe

In reference to PAPI Learner conformance, the following terms are defined in the context of protocol conformance: a "supported" feature is one that may be used by any application of the PAPI learner protocol; a feature is "used" if it is read, written, or operated upon by an application of the PAPI learner protocol; a feature is "tested" if an application of the PAPI learner protocol inquires about the existence of said feature; a feature is "accessed" if an application of PAPI learner protocol attempts to read or write data associated with the feature; a feature is "probed" if an application of the PAPI learner protocol implicitly tests the existence of a feature by attempting to *use* the feature (see "use" above) within a "safe" environment that does not cause undefined behavior.

Note: Protocol conformance makes requirements on both the protocol binding and on applications that use the protocol binding, i.e., conformity assessment of a PAPI implementation based on protocol conformance is determined by proper definition of the protocol and proper use of the protocol.

4.6 Data Application conformance

Data application conformance is measured by how well the data application behaves according to this Standard.

There are two types of data application conformance: strictly conforming and conforming.

4.6.1 Strictly conforming data application

For all strictly conforming data applications,

- Mandatory features shall exist (or shall be available) and shall conform to this Standard.
- Optional features may exist (or may be available) and, if they exist (or are available), shall conform to this Standard.
- Extended features shall not be directly used and shall not be tested for existence or availability. Note: A strictly conforming application might indirectly use an extended feature because that feature is hidden within an implementation; see definition of "consume" in Clause 3, Definitions, for this special case.

Note: A strictly conforming data application *may be minimally conforming but is maximally interoperable with respect to this Standard.* Strict conformance concerns (1) the assessment, measurement, and/or availability of a minimal set of features; (2) the data application's non-use of feature-probing; and (3) the data application's non-use of extended feature sets.

4.6.2 Conforming data application

For all conforming data applications,

- Mandatory features shall exist (or shall be available) and shall conform to this Standard.
- Optional features may exist (or may be available) and, if they exist (or are available), shall conform to this Standard.
- Extended features may exist (or may be available), may be tested for existence (or availability), and their use and behavior shall be implementation-defined.

Note: A conforming data application *may be more useful, but may be less interoperable with respect to this Standard.* Conformance concerns (1) the assessment, measurement, and/or availability of a minimal set of features; (2) feature-probing for and/or prior agreement to the existence (or availability) of extended features, as permitted by the implementation; and (3) extended features specified external to this Standard.

4.6.3 Both strictly conforming and conforming data applications

This subclause is informative and not normative.

4.6.3.1 Conformity assessment

Although all strictly conforming data applications are also conforming data applications, the *conformity assessment* of strictly conforming data applications may differ from the *conformity assessment* of conforming applications. The requirement that a data application must be both "a strictly conforming data application" and "a conforming data application", is a stronger requirement than the individual requirements of "a strictly conforming data application" and "a conforming data application", i.e., from the perspective of conformity assessment, an application may be "strictly conforming", "conforming", both, or neither.

4.6.3.2 Illustrations

It is possible for a conforming data application to be simultaneously (1) a strictly conforming data repository, (2) a conforming data repository, and (3) a generator and/or interpreter of data extensions — which appears to be in conflict with the nature of strictly conforming implementations. The following examples show two difference implementation strategies. These examples use data repositories for illustration, but the illustration applies also to data readers and data writers.

Example 1: Data repository P uses an implementation strategy that allows an *arbitrary* set of data element identifiers to be stored and retrieved (in additional to those described in this standard). When data is stored into and retrieved from P, P uses a particular binding data that data extensions are permitted to be ignored, e.g., as a coding binding that uses "extension prefixes", an API binding that "hides implementation details", or a protocol that uses "fallback negotiation and out-of-band messages". P will consume and interpret all strictly conforming data. P generates and produces strictly conforming data because the particular binding chosen "hides" the extensions; thus, all strictly conforming data readers can consume and interpret data from P. P can store extensions (although no claims are made about the specification and validity of extensions). Thus, (1) P is a strictly conforming data repository, (2) P is a conforming data repository that can store and retrieve extensions, (3) P can generate and interpret data extensions, (4) P is both a strictly conforming and a conforming data repository.

Example 2: Data repository Q uses an implementation strategy (1) that closely parallels this Standard, and (2) would be described as "minimalist". Q uses the same techniques as P does for consuming, interpreting, generating, and producing data. However, Q uses a "bucket" to store all data extensions. The "bucket" approach may have strengths (e.g., simplicity) and weaknesses (e.g., store/retrieve performance are poor) when compared to other implementations. Q also satisfies the same three requirements as P does (Q is a strictly conforming data repository; Q is a conforming data repository that can store and retrieve extensions; Q can generated and interpret data extensions) and has the same conclusion: Q is both a strictly conforming and a conforming data repository.

4.6.4 Data application varieties

There are three types of strictly conforming/conforming data applications: data repository, data reader, data writer.

Rationale

There are three separate application conformance perspectives: data repository, data reader, data writer. Vendors and administrators or data repositories will want to claim conformance ("My PAPI learner repository conforms to the Standard"). Vendors will want to claim conformance for their tools (data readers: "My application imports data and is a conforming PAPI learner data reader", data writers: "My application exports data and is a conforming PAPI learner data writer", or both). See 4.4, Data Instance Conformance, above, for additional perspectives on conformance.

4.6.5 Data repository

A data repository is a data application that stores and retrieves data objects.

A strictly conforming data repository shall:

1. receive data sets for subsequent retrieval;
2. use strictly conforming data interpretation for receiving data sets;
3. store data sets in persistent storage so that data extensions may not persist;
4. send, on request, previously stored data sets;
5. use strictly conforming data generation for sending data sets; and
6. strictly conform to at least one PAPI learner coding binding and at least one PAPI learner API or PAPI learner protocol binding.

Note 1: A strictly conforming data repository does not require "preservation" of extended data elements, i.e., data interchange should not be dependent upon expecting extended data elements to persist in a strictly conforming data repository but does not prohibit it either. See subclause 4.6.3, Both Strictly Conforming and Conforming Data Applications, for more information on the storage of extensions in strictly conforming data repositories.

A conforming data repository shall:

1. receive data objects for subsequent retrieval;
2. use conforming data interpretation for receiving data sets;
3. store data sets in persistent storage so that data extensions may persist;
4. send, on request, previously stored data objects;
5. use conforming data generation for sending data sets;
6. conform to at least one PAPI learner coding binding and at least one PAPI learner API or PAPI learner protocol binding.

Note 2: A conforming data repository may, upon storage, add, delete, or change extended data elements for subsequent retrieval.

Note 3: A conforming data repository may store *some* data extensions, but it is not required to store and retrieve *all* data extensions.

Note 4: A conforming data repository may store and retrieve data objects that are not data sets.

4.6.6 Data reader

A data reader is a data application that operates as if it (1) consumes data, and (2) interprets data which results in data sets.

Note 1: The "as if" rule implies that, conceptually, the data reader processes the information in two phases (consumption and interpretation), but the design of implementations are not constrained and implementations may use any number of phases of processing.

A strictly conforming data reader shall interpret data that strictly conforms to (1) this Standard, and (2) at least one binding of this Standard.

Note 2: A strictly conforming data reader does not interpret extended data elements.

Note 3: Depending upon the binding of this Standard, a strictly conforming data reader may "ignore" data extensions, e.g., a strictly conforming data reader may consume data extensions but the data reader is able to ignore (not interpret) these extensions.

A conforming data reader shall interpret data that conforms to (1) this Standard, and (2) at least one binding of this Standard.

Note 4: A conforming data reader may interpret extended data elements.

4.6.7 Data writer

A data writer is a data application that operates as if it (1) generates data from data sets, and (2) produces data.

Note 1: The "as if" rule implies that, conceptually, the data writer processes the information in two phases (generation and production), but the design of implementations is not constrained and implementations may use any number of phases of processing.

A strictly conforming data writer shall generate data that strictly conforms to (1) this Standard, and (2) at least one binding of this Standard.

Note 2: A strictly conforming data writer does not generate extended data elements.

A conforming data writer shall generate data that conforms to (1) this Standard, and (2) at least one binding of this Standard.

Note 3: A conforming data writer may generate extended data elements.

5 Functionality

Note : This Clause refers to "applications", which refer to information technology systems or components that use or incorporate PAPI Learner. The term "data application" is a specialized term whose requirements are specified within Clause 4, Conformance.

5.1 System functional requirements

A PAPI learner implementation shall (1) represent human information for use by learning technology systems; or (2) communicate this information among PAPI learner applications.

This Standard describes interfaces and a framework of human information repositories that are, conceptually, partitioned according to their applications' use and their administration.

PAPI learner implementations may affect the transfer of data via:

- PAPI learner application programming interfaces (APIs);
- PAPI learner protocols; or
- methods outside this Standard.

PAPI learner codings and data formats may be used to encode the data for any of the above methods.

Note: Implementations need not support

5.2 Application requirements synopsis

This subclause is informative and not normative.

The following applications are both representative and prototypical — they illustrate the required functionality of PAPI Learner.

Note 1: These applications are useful for validating the PAPI learner functionality and are not required by this Standard.

Note 2: The following single-letter abbreviations are used for PAPI learner information types: N = personal, R = relations, S = security, M = preference, G = performance, W = portfolio.

Note 3: A "✓" symbol means that this information type is <u>*most likely used*</u> in the application or requirements synopsis. A "?" symbol means that this information type <u>*might be used*</u> in the application or requirements synopsis.

Application/Requirements Synopsis	**N**	**R**	**S**	**M**	**G**	**W**

Application/Requirements Synopsis	N	R	S	M	G	W
A web-based, local area network, standalone, or nomadic application that accesses the learner's history to determine the most productive learning experience.				?	✓	
A web-based, local area network, standalone, or nomadic application that records bookmarks, lesson grades, and commentary.				?	✓	?
A learning technology application that combines the components of content delivery, management, and recordkeeping into a single application.		✓	?	✓	✓	?
A learning technology application that separates the components of content delivery, management, and recordkeeping into individual, separate applications.		✓	?	✓	✓	?
A learning technology application that uses learner histories from more than one repository (distributed repositories of learner information).		?	?	?	✓	
A learning technology application that uses learner information from multiple repositories with separate, independent security administration systems.		?	?	?	✓	
A learning technology application on a remote campus that uses learner history from the home campus.		✓	?	?	✓	?
A learning technology application on a remote campus that records learner history at the home campus.	?			?	✓	
A learning technology application that adapts to a learner's physical limitations, e.g., blindness, deafness.		?	?	✓	✓	
A learning technology application that adapts to the learner's culture, e.g., language, conventions, units of measurement, currency.		?		✓		
A back office system that transfers information between the registrar and the profile repositories.		✓	✓	✓	✓	?
An application that transfers grades from one institution to another institution in a secure environment (confidential, tamper-proof).	?			?	✓	?
An employment system that offers employees resumes, works, and accomplishments to prospective employers.						✓
An employment system that searches for learners with appropriate skills.						✓
An information structure that supports grouping of individual learners.		✓	✓	?		
A learning content development application that correlates learner performance to learner content effectiveness.		?		?	✓	
An application that collects information on the effectiveness of courses.		?	?	?	✓	?
An application that determines if a learner is enrolled in an institution and/or has paid the required monies.	✓	✓	?			
An application that prints report cards.	✓				✓	

Application/Requirements Synopsis	N	R	S	M	G	W
An application that prints mailing labels for all learners in an institution.	✓					
An application that provides a campus directory.	✓					
An application that provides emergency contact information.	✓	?				
Integration of applications in various environments: personal computer, workstation, scripting languages, compiled languages, operating systems, content delivery systems, database systems, network systems, nomadic systems, security systems, learner identification systems, administration systems.	✓	✓	✓	✓	✓	✓

5.3 Technical requirements synopsis

This subclause is informative and not normative.

The following are technical requirements for PAPI Learner.

5.3.1 Controlling access to information

Much human information is coded, stored, sorted, selected, and finally, distributed to computer programs, learners, parents, teachers, administrators, and institutions. For privacy, regulatory, technical, and other reasons:

- **Some information is kept private.** Example: learner personal information.
- **Some information has restricted access.** Example: learner performance information, learner relations information, learner portfolio information.
- **Some information has varying degrees of public accessibility.** Example: learner preference information, learner relations information, learner security information.
- **Some information must be available to certain application components.** Example: learner performance information, learner relations information, learner security information, learner preference information.
- **Some information is available to administration and management components.** Example: learner personal information, learner relations information, learner security, learner performance information.
- **Some information is primarily useful for humans.** Example: learner portfolio information.

Applications may have further constraints:

- **Applications may have access to too much information.** An application that only needs access to learner performance information (e.g., tutoring and training systems) doesn't need access to personal information, such as home phone numbers, parent names, and ethnicity information. In fact, some regulations and business practices require that *all* records be completely de-identified (no name, institution, age, etc.) with only an employee number (student number, patient number, etc.) to track progress. Example: A learner uses courseware over the internet at some distance learning center; the courseware might need access to a learner's history to determine the best lesson plan, but why does this courseware need to know private information, such as the names of the learner's parents? *Note: Other industries, such as, telemedicine (medical) and electronic commerce (financial) applications, may have access to too much information or may have potential business conflicts in jointly developed applications (e.g., in a given application, if there is only one combined repository for learner and patient preferences, does a learning technology application have access to patient information for this application?).*

- **Applications may want to store *their* information in *their* repository.** Some applications prefer to store their information in their own repositories. For example, a learner purchases a distance learning course from an institution, but the institution doesn't permit the transfer of records until the learner has completed his/her final tuition payments. *Note: In other industries, doctors, hospitals, and financial institutions will each want to maintain their own copy of certain information.*

While information may be kept in separate repositories, there also exists the need to access human information *as if it were* in a single, consolidated, combined repository (or singular database).

Example: An application that only wishes to access the learner performance information (or medical clinical information) of a combined repository may specify different "views", which imply different subsets. In this example, views are organized hierarchically: each user has a view; this view has education, healthcare, and financial sub-views; the education view has personal, preference, performance, and portfolio sub-views. Each view

```
personal.edu.john.doe
relations.edu.john.doe
security.edu.john.doe
preference.edu.john.doe
performance.edu.john.doe
portfolio.edu.john.doe
personal.medical.john.doe
demographic.medical.john.doe
clinical.medical.john.doe
financial-aid.financial.john.doe
credit-history.financial.john.doe
```

has its own security, access, and permissions, but these logical views access a combined repository (a single, common database). These separately controlled views give the appearance of separate repositories, yet might actually refer to the same repository.

Note: The naming of views may use a different hierarchical ordering, such as the following:

```
/doe/john/edu/personal
/doe/john/edu/relations
/doe/john/edu/security
/doe/john/edu/preference
/doe/john/edu/performance
/doe/john/edu/portfolio
/doe/john/medical/personal
/doe/john/medical/demographic
/doe/john/medical/clinical
```

```
/doe/john/financial/financial-aid
/doe/john/financial/credit-history
```

The choices of designing repositories as combined or separate, distributed or centralized, on-line or nomadic, etc., are implementation details outside the scope of PAPI Learner.

5.3.2 Cultural and institutional conventions

PAPI learner information must support:

- **Cultural dependencies.** Some cultural conventions may afford automated translation (e.g., feet to meters, Fahrenheit to Celsius), while some cultural conventions may be automated but time-dependent (e.g., US dollars to Euros). Some data, such as instructor or physician comments written in English, may be useful in certain cultures and might not afford automated translation. An engineering goal in PAPI Learner is to maximize the internationalization (I18N) capabilities (i.e., implementations that are interoperable and not culturally dependent) while supporting localization (L10N) mappings to local, and cultural conventions, i.e., adaptation to local and regional cultural conventions.. Example: Date and time are stored in UTC (Coordinated Universal Time), but may be converted to the local time via common library functions (e.g., **`localtime()`** in C). *Thus, an engineering goal of PAPI Learner is to support cultural conventions in the general sense, e.g., including linguistic conventions*.
- **Institutional conventions.** Some data might not afford automatic translation among institutions, e.g., for grades, what is the mapping of numeric grades to letter grades or vice versa?. It is impractical to build cross-institutional consensus around a simple, singular coding scheme. Thus, engineering goals in PAPI Learner include (1) to supporting institutional conventions, e.g., coding schemes such as "**`US-NY-K12-LETTER-GRADE`**" for New York State public schools letter grades; (2) supporting naming conventions to minimize conflicts, e.g., XYZ University that has the domain "**`xyz.edu`**" uses the coding conventions prefix/suffix "**`xyz.edu`**"; (3) trusting institutions to reduce the number of coding schemes as they benefit from higher interoperability. Similarly, medical and financial information have similar institutional conventions. *An engineering goal of PAPI Learner is to "let the market solve the problem" of choosing/reducing the number of coding schemes to the "right" level*.

5.3.3 Connectivity to information

Applications may require both on-line and off-line access to resources:

- **Applications may be disconnected from centralized resources.** Distance learning does not imply continuous on-line access. For example, the learner may connect briefly each morning to exchange information but is otherwise disconnected from the internet or other centralized resources. These nomadic users are "sometimes-connected" and/or "sometimes-roaming". Nomadic use is required when the learner is traveling, on-line access is impractical, or continuous access is expensive.

5.3.4 Partitioning information

This Standard organizes and partitions information according to overall:

- **Application Use:** Compare learner relations information to learner performance information.
- **Administration:** Compare single-copy, private learner personal information to multiple-copy, semi-public learner preference information.

Note: It is expected that this Standard will be widely adopted for interchange and access of public and private information. To promote adoption, *an important PAPI engineering requirement is supporting simple application paradigms*. In other words, it should require minimal effort to incorporate PAPI learner codings, APIs, and protocols into existing applications.

5.3.5 Summary of engineering goals

The following is a summary of PAPI Learner engineering requirements and goals:

- Controlling access to information to the extent necessary.
- Maximizing performance of accessing data.
- Supporting varying data and information structures.
- Supporting varying cultural conventions.
- Supporting varying institutional conventions.
- Supporting varying industry conventions.
- Supporting varying coding and extension mechanisms.
- Supporting varying information partitioning schemes.
- Letting the market choose the best coding scheme(s).
- Supporting varying types of on-line, "sometimes", and off-line connectivity.
- Supporting varying geographic (nomadic) access to information.

6 Conceptual model

This section defines the conceptual model of conforming PAPI Learner implementations.

6.1 Learner information, learner profiles, PAPI Learner

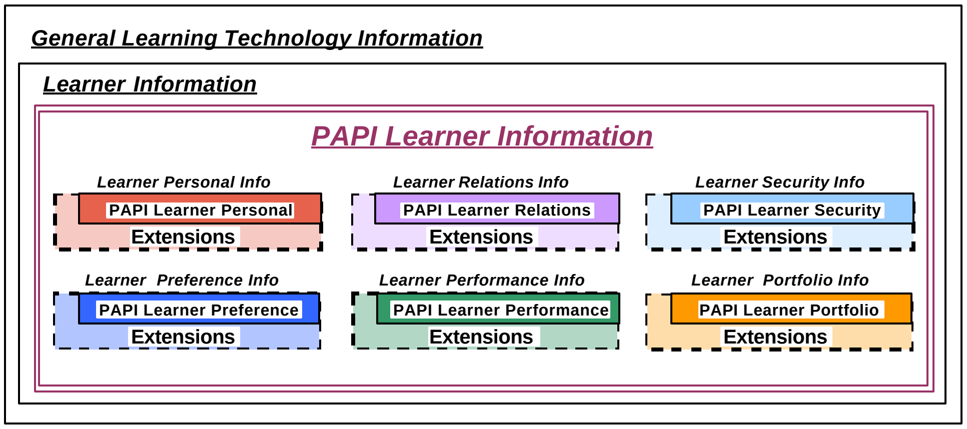

Figure 1. Relationships among general learning technology information, PAPI learner information (this Standard), and PAPI learner extensions.

Learner information, also known generically as a "learner profile", is a subset of general learning technology information. Learner information includes personal, preference, performance, portfolio, and, possibly, other types of information. This Standard describes a *particular subset* of learner information. The term "learner profile" is the generic name; PAPI learner information is one specific description of this "learner profile".

Note: For each of the six information types, this Standard describes a subset that is useful and can be widely implemented. This Standard does not describe all possible learner information, but includes only the minimum information necessary to satisfy the functional requirements and to be maximally portable, and the ability to extend this information.

The following is a brief description of the six information types of PAPI Learner:

- **Learner personal information** is not directly related to the measurement and recording of learner performance and is primarily related to administration. Note: Typically, this type of information is private and secure.
- **Learner relations information** is about the learner's relationship to other users of learning technology systems, such as teachers, proctors, and other learners.
- **Learner security information** is about the learner's security credentials, such as: passwords, challenge/responses, private keys, public keys, biometrics.
- **Learner preference information** describes preferences that may improve human-computer interactions.

- **Learner performance information** relates to the learner's history, current work, or future objectives and is created and used by learning technology components to provide improved or optimized learning experiences.
- **Leaner portfolio information** is a representative collection of a learner's works or references to them that is intended for illustration and justification of his/her abilities and achievements.

Note 1: Implementations may extend or combine these six information types. Implementations may link separate data repositories of information types via, say, database keys.

Example: A data repository of learner personal information is linked to a data repository of learner performance information by the use of a learner identifier.

Note 2: This Standard does not require these six information types to be separated, but many implementations maintain separate repositories to satisfy security, administration, regulatory, and system performance needs.

Throughout this Standard, one-letter mnemonics and colors are used as a shorthand for types of PAPI learner information:

- **N = PAPI learner personal information**. The mnemonic is "**<u>N</u>ame**". The color is red or rose.
- **R = PAPI learner relations information**. The mnemonic is "**<u>R</u>elations**". The color is violet or light violet.
- **S = PAPI learner security information**. The mnemonic is "**<u>S</u>ecurity**". The color is sky blue or light sky blue.
- **M = PAPI learner preference information**. The mnemonic is "**<u>M</u>y configuration**". The color is deep blue or pastel blue.
- **G = PAPI learner performance information**. The mnemonic is "**<u>G</u>rades**". The color is deep green or pastel green.
- **W = PAPI learner portfolio information**. The mnemonic is "**<u>W</u>orks**". The color is goldenrod or light goldenrod.

Rationale overview for information types

The following requirements motivate the need for at least six types of learner information. IEEE 1484.2.2, PAPI Learner Rationale, provides more information.

Each of the following topics represents a continuum of perspectives, e.g., there isn't only "local" and "remote" information, but many gradations and variations.

- **Local vs. Remote.** Learner information can vary in "distance" from the learner. Local information, typically, is characterized by on-line availability, higher performance access, and fewer security restrictions. Remote information, typically, is characterized by sometimes-availability, lower performance access, and significant security restrictions. Example: Compare a home campus PAPI server to a remote campus PAPI server.
- **Private vs. Public.** Learner information has varying degrees of privacy and public availability. This Standard is largely motivated by privacy issues for users, institutions, and the internet. A primary concern is: what information is available and to

whom? The PAPI Learner security paradigm is directed by a "need to know". Compare PAPI learner personal information (typically, private) to PAPI learner preference information (often, public); or, compare both (restricted) public and (restricted) private performance information.

- **Learning Experiences vs. Other Uses.** Learner information can be separated into information that is mostly useful or required for learning technology systems to give improved or optimized learning experiences (e.g., the learner's history), and all other information. Improved or optimized content sequencing is one technique for improving the learning experience. Compare learner performance information (useful or required for better learning experiences) to learner personal information (typically not required to improve learning experiences).
- **Content-Generated vs. Learner-Generated.** Learner information can be generated by the learning technology systems, generated by the learner, generated by both, or generated by others, or some combination. Compare learner performance information (generated by using learning content) to learner portfolio information (typically created and maintained by the learner).

6.2 Information types vs. data repositories

A data repository may hold one or more information types. A data set may hold one or more information types.

Example 1: Separate Data Repositories: A conforming implementation might have one data repository for each information type: a learner personal information repository, a learner relations information repository, a learner security information repository, a learner preference information repository, a learner performance information repository, a learner portfolio information repository.

Example 2: Combined Data Repository: A conforming implementation might have one repository for all types of learner information. In this example, the security administrator might allow unrestricted access to "learner preference fields" (i.e., learner preference information), while restricting access to "learner personal fields" (i.e., learner personal information); thus learner personal information is "private" and learner preference information is "public", yet they are both accessed via the same repository. The application or user might be unaware that the same data repository is accessed for each information type.

6.3 Data access model

A PAPI learner data set is a collection of information about a learner in a learning technology system. This information may include one or more types of learner information: personal, relations, security, preference, performance, portfolio.

Data interchange of PAPI learner data sets is affected by:

- **PAPI learner codings**. A PAPI Learner data set may be coded in a PAPI Learner coding binding. The data interchange is facilitated among data exchange participants by mutual agreement via methods external to the PAPI Learner coding specification.

- **PAPI learner application programming interfaces (APIs)**. The API binding is a control transfer mechanism (control is transferred from caller to callee) that affects data interchange.
- **PAPI learner protocols**. The protocol binding is both a control transfer mechanism and a data transfer mechanism.

Collectively, these bindings are called "PAPI learner codings, APIs, and protocols".

Note: PAPI learner data applications may use one or more PAPI learner codings, APIs, and protocols.

The following is the conceptual model of data access:

- **Data Object Model.** A data object shall be at least one of: a data element, or an implementation-defined object.
- **Data Storage Model.** Data, including data sets, may be stored in a data object, as referenced by an identifier.
- **Data Retrieval Model.** Data, including data sets, may be retrieved from a data object, as referenced by an identifier.
- **Data Typing Model.** Data objects that are data elements shall have a datatype. Datatypes may prescribe certain value spaces (e.g., domains), representation, encoding, storage, layout, conversion to other types, methods, and operations. The datatype of PAPI Learner data elements is defined by this Standard, which uses the semantics and notation of ISO/IEC 11404.
- **Data Attribute Model.** A data attribute shall be an implementation-defined object associated with a data object. These attributes themselves may be accessed as data objects. Note: Attributes are also known as "properties".
- **Data Repository Access Model.** PAPI Learner bindings define access, if any, to data repositories.
- **Data Repository Security Model.** See subclause 6.8, Security Model, below.
- **Data Persistence Model.** The lifetime of data objects shall be implementation-defined.
- **Data Navigation Model.** The techniques for navigating data structures are defined in PAPI Learner bindings.
- **Data Identification Model.** The identification, labeling, namespace, and their associated techniques shall be implementation-defined.
- **Data Referencing Model.** A data repository may create a reference to a data object for the purpose of subsequent dereference. The naming conventions, lifetime, and scoping of a reference shall be implementation-defined.
- **Data Dereferencing Model.** A data repository may access a data object based upon supplying a reference, i.e., dereferencing a reference. The dereferencing methods shall be implementation-defined.
- **Data Indexing Model.** The indexing methods for data repositories shall be implementation-defined. Note: The term "indexing" is used in the context of database systems, i.e., methods for organizing database records.
- **Data Searching Model.** The searching methods for data repositories shall be implementation-defined.

6.4 Extending PAPI within an application area

The specification of extended features is outside the scope of this Standard.

Note: Even though information technology components may support a wide variety of implementations, data repository administrators may choose to include and/or limit certain extensions. See IEEE 1484.14.x "Extensions Techniques", for more information.

6.5 Limiting PAPI extensions

The specification or limitation of these extended features is outside the scope of this Standard.

Note: Even though information technology components may support a wide variety of implementations, data repository administrators may choose to limit the availability or use of extensions. See IEEE 1484.14.x "Extensions Techniques", for more information.

6.6 Distance, distributed, and nomadic systems

Distance features are supported by permitting global namespaces and the infrastructure to search, store, and retrieve information within a global namespace. The namespace conventions and resolution methods shall be implementation-defined.

Distributed features are permitted by a single repository storing one or more information types; and by one ore more repositories acting as a single repository. The synchronization, replication, commit, rollback, and fallback methods shall be implementation-defined.

Nomadic features are permitted by roaming users within a sometimes-connected infrastructure. The methods of dynamic quality of service and continuity of connection shall be implementation-defined.

6.7 Correlation of granularity levels

The correlation of various levels of granularity for learner performance information shall be implementation-defined.

Example: If a teacher changes a course grade (smaller granularity), when does the grade point average (larger granularity) automatically change?

6.8 Security model

Security is defined and bounded by a security perimeter. The following features shall be implementation-defined:

- The boundary of security perimeter(s).
- The nature, type, and acceptable level of risk of inbound security threats.

- The nature, type, and acceptable level of risk of outbound security threats.
- The security strength.
- The parameterization, setup, negotiation, and knockdown of security features.
- The administration of the security perimeter integrity.

The interoperability agreements are defined externally to this Standard.

Note: The Security Model is harmonized with ISO/IEC 17799-1 "Code of Practice for Information Security Management".

The following security features are defined in the conceptual model:

- **Session-View-Based Security Model.** Security features are provided on a per-session, per-view basis. Each security session is initiated by an accessor (a user or agent that requests access). The accessor provides security credentials that authenticate the accessor, authorize the accessor, or both. A view represents a portion of PAPI learner information; a "view" is similar to the notion of a database "view". Each view established represents a session, i.e., the "session" represents the duration of access and the "view" represents the scope of access.
- **Security Parameter Negotiation Model.** Data interchange participants negotiate security parameters prior to, during, and after each session. The security parameters are defined in the PAPI Learner bindings.
- **Security Extension Model.** Additional security features may be used that were not foreseen. The method of incorporating security extensions is defined in the PAPI Learner bindings.
- **Access Control Model.** Accessors may attempt read access upon data elements, may attempt write access upon data elements, may attempt to create new data elements (separately or within aggregates), may attempt to destroy data elements (separately or within aggregates), and/or may attempt to change attributes of data elements. Other access methods, if any, shall be implementation-defined.
- **Identification Model.** The methods for identifying learners shall be implementation-defined. Note: IEEE 1484.13, Simple Human Identifiers, define the datatype associated with a learner identifier.
- **Authentication Model.** The methods of authenticating users are outside the scope of this Standard.
- **De-identification Model.** All information, except learner personal information, should be de-identified. The methods of de-identifying learners and their information are outside the scope of this Standard. Note: Administrators and, possibly, learners are responsible for choosing appropriate learner identifiers that do not reveal the learner's identity. Example: Choosing a learner's name as his/her identifier is a poor choice from the perspective of de-identification. A variety of better de-identification techniques are possible, including the use of random numbers with short "lifetimes".
- **Authorization Model.** The methods of authorizing operations shall be implementation-defined.
- **Delegation Model.** The methods of delegating administration, authority, or credentials shall be implementation-defined.
- **Non-Repudiation Model.** The methods of non-repudiation shall be implementation-defined.

- **Repudiation Model.** The methods of repudiating data, users, or credentials shall be implementation-defined.
- **Privacy Model.** Note: This Standard specifies neither a privacy model nor privacy requirements, but supports security frameworks and approaches that permit the implementation of a wide variety of privacy frameworks.
- **Confidentiality Model.** This Standard specifies neither a confidentiality model nor confidentiality requirements, but supports access controls and the partitioning of information types that permit the implementation of a wide variety of confidentiality frameworks.
- **Encryption Model.** This Standard specifies neither an encryption model nor encryption requirements, but supports several security frameworks and techniques that permit the integration of various encryption models and technologies.
- **Data Integrity Model.** This Standard specifies neither a data integrity model nor data quality requirements, but supports information assurance frameworks and approaches that permit the implementation of a wide variety of data integrity frameworks
- **Validation of Certificates.** This Standard does not require validation of learner performance information or learner portfolio information, but supports the parameterization of automated validation of either type of learner information.
- **Digital Signature Model.** This Standard specifies neither a digital signature model nor digital signature requirements, but supports several signing frameworks and techniques permit the integration of various digital signature models, policies, and technologies. This Digital Signature Model is harmonized with ISO/IEC 15945 "Specification of Trusted Third Party Services to Support the Application of Digital Signatures".

IEEE 1484.2.3, PAPI Learner Information Security Notes, contains information about applying security techniques and technologies to PAPI Learner implementations.

7 Semantics

The meaning of PAPI learner information shall be the same regardless of which PAPI Learner binding and PAPI Learner encoding is chosen.

Note 1: The choice of PAPI Learner bindings and/or PAPI Learner encodings is for the convenience of the data interchange participants.

Note 2: PAPI learner coding bindings define a mapping for each datatype defined in this Clause.

Note 3: PAPI learner API bindings define a mapping for each operation and datatype defined in this Clause.

Note 4: PAPI learner protocol bindings define a mapping for each operation and datatype defined in this Clause, and define methods for setup and knockdown of supporting communication networks.

Note 5: Throughout this Clause, the abbreviate SPM is used, which means "smallest permissible maximum". The SPM value is intended to give implementers a lower limit on conforming implementations. Applications should not assume that implementations support capabilities beyond the SPM value unless prior arrangements have been made.

7.1 General data operations

PAPI learner information shall support the following data management operations on data sets:

- **Create Operation.** Creating a new instance of some information type, such as personal information.
- **Destroy Operation.** Discarding an instance of an information type in the context of its storage. Note: Compare destroying a record in application memory, in temporary storage, and in a database.
- **Copy Operation.** Creating a new instance of an information type with identical contents.
- **Move Operation.** Changing a label associated with an instance of an information type by changing the storage of the information (implicit label change) or changing the label itself (explicit label change). Example: An implicit label change might be affected by creating new "hard links" to a new label, then deleting "hard links" to the old label. An explicit label change might be affected by changing the label in some "directory" of information.
- **Label Operation.** Creating (or removing) a name, specified by the "caller", to be associated with an instance of information.
- **Navigate Operation.** Using a naming method (absolute, relative, complete, progressive) to locate an instance of an information type.

- **Search Operation.** Finding instances of an information type that match search criteria and returning the found information via references, labels, or copies.
- **Reference Operation.** Creating a handle to an instance of an information type. Note: The difference between a label and a reference is: the "caller" chooses the name for a label, while the "callee" chooses the name for a reference.
- **Dereference Operation.** Using a handle, created through reference, to access an instance of an information type.
- **Aggregation Operation.** Combining several instances of one or more information types into a single container.
- **Decomposition Operation.** Extracting instances of information types from a container.

The PAPI Learner binding defines the methods for accessing data management operations and the availability of said operations.

7.2 Application-specific data operations

PAPI learner information shall support the following application-specific data operations on data sets:

- **Accumulation Operation.** PAPI learner records may be accumulated, aggregated, or analyzed. Examples: "what it the average score among third graders?", "what is the grade point average for a particular learner?"
- **Time Compression and Expansion Operations.** PAPI learner records may be recorded at various levels of granularity. Time compression reduces the set of records to larger granularity. Time expansion creates records of finer granularity by interpolation. Example: quarterly grades and a final exam are rolled up into a final grade; after compression, the data set is reduced because only the final grade remains.
- **Sort Operation.** PAPI records may be ordered, based upon sort criteria. Example: users may be ordered alphabetically by name.

The PAPI Learner binding defines the methods for accessing application-specific data operations and the availability of said operations.

7.3 Data compatibility

PAPI learner information may support the following data compatibility operations on data sets.

- **Promotion of Data Types.** Transforming a value from lesser capabilities to greater capabilities. Example: Converting an integer to a real; converting a character array of length 10 to a character array of length 20; converting a record of elements **`{Name, Address}`** to a record of a superset of elements **`{Name, Address, Telephone}`**.
- **Demotion of Data Types.** Transforming a value from greater capabilities to lesser capabilities. Example: Converting a real to an integer; converting a character array of length 20 to a character array of length 10; converting a record of elements **`{Name, Address, Telephone}`** to a record of a subset of elements **`{Name, Address}`**.

- **Conversion To/From Data Types.** Transforming a value from one data type to another data type. Example: Converting an integer to a string; converting the pair `{Primary-Name, Secondary-Name}` to `{Sort-Name}`.
- **Text Formats.** Transforming values to/from textual representations.

The PAPI Learner binding defines the methods for accessing data compatibility operations and the availability of said operations.

7.4 Foundational datatypes

The following datatypes are used by more than one PAPI learner information type.

In this subclause and the remaining subclauses of this Clause, the ISO/IEC 11404 summary provides additional information, such as size and smallest permitted maximum values, if not already provided in the Description wording.

7.4.1 PAPI_learner_context_label_type

ISO/IEC 11404 summary

```
type PAPI_learner_context_label_type =
    characterstring(iso-10646-1),  // SPM: 500
```

Description

All sizes or limits are smallest permitted maximum values.

A `characterstring` of size 500. The meaning of the values shall be implementation-defined.

Example

The value `"!(japan)"` might be "not within the context of Japan" — whatever that means for the implementation.

7.4.2 arraylist

ISO/IEC 11404 summary

```
type arraylist(type_spec,size) =
    array (0..size-1) of (type_spec),
```

Description

A shorthand for multiple array elements. This type declaration is only a "synonym" (in the ISO/IEC 11404 sense of a `"new"` type) for an existing type.

7.4.3 mlstring_type

ISO/IEC 11404 summary

```
type mlstring_type =
record
(
    string:
        characterstring(iso-10646-1),  // SPM: 1000
    locale:
        characterstring(iso-646),  // SPM: 255
),
```

Description

A multilingual string. The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`string:`** The localized string.
- **`locale:`** The localization (L10N) mapping, known as the "locale".

Example

The following are sample ISO/IEC 11404 (data set) values:

```
// ISO/IEC 11404 data set
(
    string = "hello world",
    locale = "en-US",
),
```

7.4.4 mlstring_array

ISO/IEC 11404 summary

```
type mlstring_array_type(limit) =
    array (0..limit-1) of (mlstring_type),
```

Description

An array of multilingual strings, also known as a "message catalog features". This type declaration is only a "synonym" (in the ISO/IEC 11404 sense of a **`"new"`** type) for an existing type.

7.4.5 PAPI_learner_bucket_type

ISO/IEC 11404 summary

```
type PAPI_learner_bucket_type =
record
(
    name:
        characterstring(iso-10646-1),  // SPM: 200
    value:
        octetstring, // SPM: 4096
),
```

Description

A "bucket" for adding name-value pairs to a PAPI Learner information type. The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`name:`** The name portion of the name-value pair.
- **`value:`** The value portion of the name-value pair.

Note: This feature permits a limited extension capability that works on all strictly conforming systems.

Example

The following are sample ISO/IEC 11404 (data set), XML (data instance), and DNVP (data instance) values:

```
// ISO/IEC 11404 data set
(
    name = "special_parameter_1",
    value = "xyz",
),

<!-- XML data instance ("..." is replaced by outer tags) -->
<...>
    <name>special_paramter_1</name>
    <value>xyz</value>
</...>

#### DNVP data instance ("..." is replaced by outer context)
....name: special_parameter_1
....value: xyz
```

7.4.6 PAPI_learner_identifier_type

ISO/IEC 11404 summary

```
type PAPI_learner_identifier_type =
record
(
    context_label :
        PAPI_learner_context_label_type,
    identifier_type :
        PAPI_learner_identifier_type_type,
    identifier_value :
        octetstring, // SPM: 1024
),
```

Description

An internal identifier that is used to link across databases. The meaning , namespace, scoping, and resolution of this feature shall be implementation-defined.

The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`context_label:`** The context for use of this data element.
- **`identifier_type:`** The type of the identifier.
- **`identifier_value:`** The value of the identifier.

Example

The following are sample ISO/IEC 11404 (data set), XML (data instance), and DNVP (data instance) values:

```
// ISO/IEC 11404 data set
(
    context_label = "database_pool_4",
    identifier_type = "pointer",
    identifier_value = "0x12345678",
),

<!-- XML data instance ("..." is replaced by outer tags) -->
<...>
    <context_label>database_pool_4</context_label>
    <identifier_type>pointer</identifier_type>
    <identifier_value>0x12345678</identifier_value>
</...>

#### DNVP data instance ("..." is replaced by outer context)
....context_label: database_pool_4
....identifier_type: pointer
....identifier_value: 0x12345678
```

7.4.7 PAPI_learner_hid_type

ISO/IEC 11404 summary

```
type PAPI_learner_hid_type =
record
(
    context_label :
        PAPI_learner_context_label_type,
    identifier_type :
        PAPI_learner_identifier_type_type,
    identifier_value :
        octetstring, // SPM: 1024
),
```

Description

An external identifier that is used to correlate PAPI learner information across repositories. The meaning , namespace, scoping, and resolution of this feature shall be implementation-defined.

The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`context_label:`** The context for use of this data element.
- **`identifier_type:`** The type of the identifier.
- **`identifier_value:`** The value of the identifier.

Example

The following are sample ISO/IEC 11404 (data set), XML (data instance), and DNVP (data instance) values:

```
// ISO/IEC 11404 data set
(
    context_label = "home",
    identifier_type = "IEEE_1484.13",
    identifier_value = "00112233",
),

<!-- XML data instance ("..." is replaced by outer tags) -->
<...>
    <context_label>home</context_label>
    <identifier_type>IEEE_1484.13</identifier_type>
    <identifier_value>00112233</identifier_value>
</...>

#### DNVP data instance ("..." is replaced by outer context)
....context_label: home
....identifier_type: IEEE_1484.13
....identifier_value: 00112233
```

7.4.8 PAPI_learner_identifier_type_type

ISO/IEC 11404 summary

```
// This datatype describes the identifier type (URL, key,
// pattern, etc.).
type PAPI_learner_identifier_type_type =
    octetstring, // SPM: 256
```

Description

The varieties of PAPI learner identifiers are describe by this datatype. The useful values are specified external to this Standard.

7.4.9 PAPI_learner_data_certification_type

ISO/IEC 11404 summary

```
type PAPI_learner_data_certification_type =
record
(   // note: all components are optional; all sizes are SPM
    certification_source : // who certified the record
        octetstring, // SPM: 2048
    certification_method : // how it was certified
        octetstring, // SPM: 1024
    certification_parameter_list :
        octetstring, // SPM: 16384
    certification_subset : // which data elements were certified
        octetstring, // SPM: 1024
    certification_identifier : // ID associated with cert. event
        octetstring, // SPM: 2048
    certification_bucket : // other information
        arraylist(PAPI_learner_bucket_type,100),
),
```

Description

PAPI learner data certification information is defined by this datatype. This datatype is used by the PAPI learner performance and PAPI learner portfolio information types.

The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`certification_source:`** Who certified this record.
- **`certification_method:`** What certification method was used.
- **`certification_parameter_list:`** The options and parameters necessary to validate the certification. Notes: With the **`certification_source`**, **`certification_method`**, and **`certification_parameter_list`**, it is possible to validate the **`certification_identifier`**.
- **`certification_subset:`** A list of which elements in this record are certified. Note: This is necessary for generating automatic validation.
- **`certification_identifier:`** The identifier that is passed to the certification validator, i.e., the certificate ID.
- **`certificate_bucket:`** A "bucket" for adding name-value pairs that provides limited extension capabilities to PAPI learner data certificate information.

7.5 PAPI learner personal information datatypes

The following datatypes are used by the PAPI learner personal information type.

7.5.1 PAPI_learner_personal_info_type

ISO/IEC 11404 summary

```
type PAPI_learner_personal_info_type =
record
(   // Note: All components are optional; all sizes are SPM
    my_personal_identifier_list : // database linking (key)
        arraylist(PAPI_learner_identifier_type,200),
    personal_hid_list : // HID linking (cross-repository)
        arraylist(PAPI_learner_hid_type,200),
    name_list :
        arraylist(PAPI_learner_name_type,40),
    telephone_list :
        arraylist(PAPI_learner_telephone_type,15),
    email_contact_list :
        arraylist(PAPI_learner_email_address_type,25),
    postal_address_list :
        arraylist(PAPI_learner_postal_address_type,10),
    personal_bucket :
        arraylist(PAPI_learner_bucket_type,100),
),
```

Description

PAPI learner personal information is defined by this datatype.

The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`my_personal_identifier_list:`** An internal database key for linking information.
- **`personal_hid_list:`** An external human identifier for correlating information across data repositories.
- **`name_list:`** The name of the learner.
- **`telephone_list:`** Telephone numbers associated with the learner.
- **`email_contact_list:`** E-mail addresses associated with the learner.
- **`postal_address_list:`** Postal addresses associated with the learner.
- **`personal_bucket:`** A "bucket" for adding name-value pairs that provides limited extension capabilities to PAPI learner personal information.

Example

The following are sample ISO/IEC 11404 (data set), XML (data instance), and DNVP (data instance) values:

```
// ISO/IEC 11404 data set
(
    my_personal_identifier_list =
    (
        (
            identifier_type = "pointer",
            identifier_value = "0x12345678"
        ),
    ),
    personal_hid_list =
    (
        (
            identifier_type = "IEEE_1484.13",
            identifier_value = "00112233"
        ),
    ),
    name_list =
    (
        (
            official_name =
            (
                primary = "Public",
                secondary = "Joseph Q.",
            ),
            sort_name =
                "Public, Joseph Q.",
            short_informal_name =
                "Joe",
        )
    ),
    telephone_list =
    (
        (
            context_label = "home",
            identifier_type = "voice",
            phone_number = "+1 212 555 1212"
        ),
        (
            context_label = "work",
            identifier_type = "voice",
            phone_number = "+1 212 555 1313"
        ),
        (
            context_label = "work",
            identifier_type = "fax",
            phone_number = "+1 212 555 1414"
        ),
        (
            context_label = "emergency",
            identifier_type = "voice",
            phone_number = "+1 212 555 1515"
        ),
    ),
    email_contact_list =
    (
        (
            context_label = "home",
            email_address_type = "rfc822",
            email_address = "foobar@mail.com"
        ),
        (
            context_label = "work",
            email_address_type = "rfc822",
            email_address = "foo@bar.com"
        ),
    ),
    postal_address_list =
    (
        (
            context_label = "school",
            addressee_name_given = "Joseph Q.",
            addressee_name_family = "Public",
            delivery_street_type = "Street",
            delivery_street_name = "Main",
            delivery_street_id_number = "123",
            delivery_city = "New York",
            delivery_territory = "NY",
            delivery_routing = "10001",
            delivery_country = "USA",
        ),
    ),
    personal_bucket =
    (
        (
            name = "social_security_number",
            value = "123-45-6789",
        ),
        (
            name = "payment_method",
            value = "371234567890123/200212/John Q. Public",
        ),
    ),
),

<!-- XML data instance ("..." is replaced by outer tags) -->
<...>
    <my_personal_identifier_list>
        <personal_identifier>
            <identifier_type>pointer</identifier_type>
            <identifier_value>0x12345678</identifier_value>
        </personal_identifier>
    </my_personal_identifier_list>
    <my_personal_hid_list>
        <learner_hid>
            <identifier_type>IEEE_1484.13</identifier_type>
            <identifier_value>00112233</identifier_value>
        </learner_hid>
    </my_personal_hid_list>
    <name_list>
        <name>
            <official_name>
                <primary>Public</primary>
                <secondary>Joseph Q.</secondary>
            </official_name>
            <sort_name>Public, Joseph Q.</sort_name>
            <short_informal_name>Joe</short_informal_name>
        </name>
    </name_list>
    <telephone_list>
        <telephone>
            <context_label>home</context_label>
            <identifier_type>voice</identifier_type>
            <phone_number>+1 212 555 1212</phone_number>
        </telephone>
        <telephone>
            <context_label>work</context_label>
            <identifier_type>voice</identifier_type>
            <phone_number>+1 212 555 1313</phone_number>
        </telephone>
        <telephone>
            <context_label>work</context_label>
            <identifier_type>fax</identifier_type>
            <phone_number>+1 212 555 1414</phone_number>
        </telephone>
        <telephone>
            <context_label>emergency</context_label>
            <identifier_type>voice</identifier_type>
            <phone_number>+1 212 555 1515</phone_number>
        </telephone>
    </telephone_list>
    <email_contact_list>
        <email_contact>
            <context_label>home</context_label>
            <email_address_type>rfc822</email_address_type>
            <email_address>foobar@mail.com</email_address>
        </email_contact>
        <email_contact>
            <context_label>work</context_label>
            <email_address_type>rfc822</email_address_type>
            <email_address>foo@bar.com</email_address>
        </email_contact>
    </email_contact_list>
    <postal_address_list>
        <postal_address>
            <context_label>school</context_label>
            <addressee_name_given>Joseph Q.</addressee_name_given>
            <addressee_name_family>Public</addressee_name_family>
            <delivery_street_id_number>123</delivery_street_id_number>
            <delivery_street_name>Main</delivery_street_name>
            <delivery_street_type>Street</delivery_street_type>
            <delivery_city>New York</delivery_city>
            <delivery_territory>NY</delivery_territory>
            <delivery_routing>10001</delivery_routing>
            <delivery_country>USA</delivery_country>
        </postal_address>
    </postal_address_list>
    <personal_bucket>
        <bucket>
            <name>social_security_number</name>
            <value>123-45-6789</value>
        </bucket>
        <bucket>
            <name>payment_method</name>
            <value>371234567890123/200212/John Q. Public</value>
        </bucket>
    </personal_bucket>
</...>

#### DNVP data instance ("..." is replaced by outer context)
....my_personal_identifier.__begin:
....my_personal_identifier.identifier_type: pointer
....my_personal_identifier.identifier_value: 0x12345678
....my_personal_identifier.__end:
....my_personal_hid.__begin:
....my_personal_hid.identifier_type: IEEE_1484.13
....my_personal_hid.identifier_value: 00112233
....my_personal_hid.__end:
....name.__begin:
....name.official_name.primary: Public
....name.official_name.secondary: Joseph Q.
....name.sort_name: Public, Joseph Q.
....name.short_informal_name: Joe
....name.__end:
....telephone.__begin:
....telephone.context_label: home
....telephone.identifier_type: voice
....telephone.phone_number: +1 212 555 1212
....telephone.__end:
....telephone.__begin:
....telephone.context_label: work
....telephone.identifier_type: voice
....telephone.phone_number: +1 212 555 1313
....telephone.__end:
....telephone.__begin:
....telephone.context_label: work
....telephone.identifier_type: fax
```

```
....telephone.phone_number: +1 212 555 1414
....telephone.__end:
....telephone.__begin:
....telephone.context_label: emergency
....telephone.identifier_type: voice
....telephone.phone_number: +1 212 555 1515
....telephone.__end:
....email_contact.__begin:
....email_contact.context_label: home
....email_contact.email_address_type: rfc822
....email_contact.email_address: foobar@mail.com
....email_contact.__end:
....email_contact.__begin:
....email_contact.context_label: work
....email_contact.email_address_type: rfc822
....email_contact.email_address: foo@bar.com
....email_contact.__end:
....postal_address.__begin:
....postal_address.context_label: school
....postal_address.addressee_name_given: Joseph Q.
....postal_address.addressee_name_family: Public
....postal_address.delivery_street_id_number: 123
....postal_address.delivery_street_name: Main
....postal_address.delivery_street_type: Street
....postal_address.delivery_city: New York
....postal_address.delivery_territory: NY
....postal_address.delivery_routing: 10001
....postal_address.delivery_country: USA
....postal_address.__end:
....personal_bucket.__begin:
....personal_bucket.name: social_security_number
....personal_bucket.value: 123-45-6789
....personal_bucket.__end:
....personal_bucket.__begin:
....personal_bucket.name: payment_method
....personal_bucket.value: 371234567890123/200212/John Q. Public
....personal_bucket.__end:
```

7.5.2 PAPI_learner_name_type

ISO/IEC 11404 summary

```
type PAPI_learner_name_type =
record
(   // note: all components are optional; all sizes are SPM values
    context_label : // e.g., "!(japan)"
        PAPI_learner_context_label_type,
    official_name : // e.g., { "Public" } { "Joseph" "Quincy" } - legal
        arraylist(PAPI_learner_formal_name_type,5),
    full_formal_name : // e.g., "Mr. Joseph Q. Public, PhD" name/titles
        arraylist(PAPI_learner_full_name_type,5),
    mrms_formal_name : // e.g., "Mr. Joseph Q. Public" - Mr./Ms. name
        arraylist(PAPI_learner_full_name_type,5),
    short_formal_name : // e.g., "Mr. Public" - within formal group
        arraylist(PAPI_learner_full_name_type,5),
    full_name : // e.g., "Joseph Q. Public" - in print
        arraylist(PAPI_learner_full_name_type,5),
    sort_name : // e.g., "Public, Joseph Q."
```

```
        arraylist(PAPI_learner_full_name_type,5),
    full_informal_name : // e.g., "Joe Public" - within a group
        arraylist(PAPI_learner_full_name_type,5),
    short_informal_name : // e.g., "Joe" - directly to person
        arraylist(PAPI_learner_full_name_type,5),
}
```

Description

The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`my_personal_identifier:`** An internal database key for linking information.
- **`context_label:`** The context for use of this data element.
- **`offical_name:`** The legal name of the learner. Example: The learner's name on his/her passport.
- **`full_formal_name:`** The full, formal name that includes titles.
- **`mrms_formal_name:`** A formal name that includes Mr., Ms., or equivalent titles.
- **`short_formal_name:`** A formal name that, typically, includes Mr., Ms., or equivalent titles, and the family name.
- **`full_name:`** The full name, as it would appear in public announcements.
- **`sort_name:`** The learner's name rearranged so that lexical ordering corresponds to name ordering. Example: "Last, First".
- **`full_informal_name:`** A full name within an informal setting.
- **`short_informal_name:`** A short informal name within an informal setting.

7.5.3 PAPI_learner_formal_name_type

ISO/IEC 11404 summary

```
type PAPI_learner_formal_name_type =
record
(
    primary :  // Family name(s).
        characterstring(iso-10646-1), // SPM: 70
    secondary : // Given name(s).
        characterstring(iso-10646-1), // SPM: 70
),
```

Description

A formal name is defined as a "primary identifier" and a "secondary identifier". The terms "primary" and "secondary" are defined in ICAO, IATA, UN, and ATA standards for automated passport information. The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`primary:`** The primary legal name(s) of a person.
- **`secondary:`** The secondary legal name(s) of a person.

7.5.4 PAPI_learner_full_name_type

ISO/IEC 11404 summary

```
type PAPI_learner_full_name_type =
    characterstring(iso-10646-1), // SPM: 140
```

Description

A localized name representation. All sizes or limits are smallest permitted maximum values.

Examples: "Mr. Smith", "Smith-san", "Mr. Nakabayashi", "Nakabayashi-san".

7.5.5 PAPI_learner_telephone_type

ISO/IEC 11404 summary

```
type PAPI_learner_telephone_type =
record
(
    context_label :
        PAPI_learner_context_label_type,
    identifier_type :
        PAPI_learner_identifier_type_type,
    phone_number :
        octetstring, // SPM: 50
),
```

Description

A telephone number, as defined by ITU-T [Editor's Note: Need ITU-T reference]. The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`context_label:`** The context for use of this data element.
- **`identifier_type:`** The type of phone number.
- **`phone_number:`** The phone number.

7.5.6 PAPI_learner_email_contact_type

ISO/IEC 11404 summary

```
type PAPI_learner_email_contact_type =
record
(
    context_label :
        PAPI_learner_context_label_type,
    email_address_type :
        PAPI_learner_identifier_type_type,
    email_address :
        octetstring, // SPM: 255
),
```

Description

An E-mail address, as defined by IETF RFC xxx [Editor's Note: Need IETF RFC reference]. The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`context_label:`** The context for use of this data element.
- **`email_address_type:`** The type of E-mail address.
- **`email_address:`** The E-mail address.

7.5.7 PAPI_learner_postal_address_type

ISO/IEC 11404 summary

```
type PAPI_learner_postal_address_type =
record
(   // note: all components are optional; all sizes are SPM
    context_label :
        PAPI_learner_context_label_type,

    // The addressee.  Note: An application should harmonize
    // the values of postal addressee name, title, etc., with
    // learner name components.
    addressee_title :
        characterstring(iso-10646-1), // SPM: 35
    addressee_name_given :
        characterstring(iso-10646-1), // SPM: 70
    addressee_name_family :
        characterstring(iso-10646-1), // SPM: 70
    addressee_name_suffix :
        characterstring(iso-10646-1), // SPM: 35
    addressee_occupation :
        characterstring(iso-10646-1), // SPM: 70
    addressee_function :
        characterstring(iso-10646-1), // SPM: 70
    addressee_care_of_address :
        characterstring(iso-10646-1), // SPM: 70

    // The organization.
    organization_name :
        characterstring(iso-10646-1), // SPM: 70
    organization_activity :
        characterstring(iso-10646-1), // SPM: 70
    organization_division :
        characterstring(iso-10646-1), // SPM: 70

    // The delivery address.
    delivery_street_type :
        characterstring(iso-10646-1), // SPM: 35
    delivery_street_name :
        characterstring(iso-10646-1), // SPM: 70
    delivery_street_id_number :
        characterstring(iso-10646-1), // SPM: 20
    delivery_supplementary_address :
        characterstring(iso-10646-1), // SPM: 70
    delivery_city :
        characterstring(iso-10646-1), // SPM: 35
    delivery_po_box :
        characterstring(iso-10646-1), // SPM: 20
    delivery_postcode :
        characterstring(iso-10646-1), // SPM: 20
    delivery_routing :
        characterstring(iso-10646-1), // SPM: 20
    delivery_office :
        characterstring(iso-10646-1), // SPM: 35
    delivery_territory :
        characterstring(iso-10646-1), // SPM: 35
    delivery_country :
        characterstring(iso-10646-1), // SPM: 35
),
```

Description

A postal address, as defined by ISO/IEC 11180. The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`context_label:`** The context for use of this data element.
- **`addressee_*:`** The individual addressee of the postal address.
- **`organization_*:`** The organization associated with the postal address.
- **`devliery_*:`** The delivery address.

Note: An application should harmonize the values of postal addressee name, title, etc., with the learner name components.

7.6 PAPI learner relations information datatypes

The following datatypes are used by the PAPI learner relations information type.

7.6.1 PAPI_learner_relations_info_type

ISO/IEC 11404 summary

```
type PAPI_learner_relations_info_type =
record
(   // note: all components are optional; all sizes are SPM
    my_relations_identifier_list : // database linking (key)
        arraylist(PAPI_learner_identifier_type,200),
    relations_hid_list : // HID linking (cross-repository)
        arraylist(PAPI_learner_hid_type,200),
    relationship_list : // List of individual relationships
        arraylist(PAPI_learner_relationship_type,200),
    relations_bucket : // Other information
        arraylist(PAPI_learner_bucket_type,200),
),
```

Description

PAPI learner relations information is defined by this datatype.

The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`my_relations_identifier_list:`** An internal database key for linking information.
- **`relations_hid_list:`** An external human identifier for correlating information across data repositories.
- **`relationsship_list:`** A list of individual relationships to this learner.
- **`relations_bucket_list:`** A "bucket" for adding name-value pairs that provides limited extension capabilities to PAPI learner relations information.

Example

The following are sample ISO/IEC 11404 (data set), XML (data instance), and DNVP (data instance) values:

```
// ISO/IEC 11404 data set
(
    my_relations_identifier_list =
    (
        (
            identifier_type = "pointer",
            identifier_value = "0x12345678"
        ),
    ),
    relations_hid_list =
    (
        (
            identifier_type = "IEEE_1484.13",
            identifier_value = "44556677"
        ),
    ),
    relationship_list =
    (
        (
            others_identifier_list =
            (
                (
                    identifier_type = "IEEE_1484.13",
                    identifier_value = "44556688"
                ),
                (
                    identifier_type = "IEEE_1484.13",
                    identifier_value = "44556699"
                ),
            ),
            relations_label_list =
            (
                (
                    string = "history_101_section_3",
                    locale = "en-US",
                )
            ),
            relation_to_them =
                "classmate",
        ),
    ),
),

<!-- XML data instance ("..." is replaced by outer tags) -->
<...>
    <my_relations_identifier_list>
        <relations_identifier>
            <identifier_type>pointer</identifier_type>
            <identifier_value>0x12345678</identifier_value>
        </relations_identifier>
    </my_relations_identifer_list>
    <my_relations_hid_list>
        <learner_hid>
            <identifier_type>IEEE_1484.13</identifier_type>
            <identifier_value>44556677</identifier_value>
        </learner_hid>
    </my_relations_hid_list>
    <relationship_list>
        <relationship>
            <others_identifier_list>
                <others_identifier>
                    <identifier_type>IEEE_1484.13</identifier_type>
                    <identifier_value>44556688</identifier_value>
                </others_identifier>
                <others_identifier>
                    <identifier_type>IEEE_1484.13</identifier_type>
                    <identifier_value>44556699</identifier_value>
                </others_identifier>
            </others_identifier_list>
            <relationship_label_list>
                <relationship_label LANG="en-US">
                    history_101_section_3
                </relationship_label>
            </relationship_label_list>
            <relationship_to_them>
                classmate
            </relationship_to_them>
        <relationship>
</...>

#### DNVP data instance ("..." is replaced by outer context)
....my_relations_identifier.__begin:
....my_relations_identifier.identifier_type: pointer
....my_relations_identifier.identifier_value: 0x12345678
....my_relations_identifier.__end:
....my_relations_hid.__begin:
....my_relations_hid.identifier_type: IEEE_1484.13
....my_relations_hid.identifier_value: 44556677
....my_relations_hid.__end:
....relationship.__begin:
....relationship.others_identifier.__begin:
....relationship.others_identifier.identifier_type: IEEE_1484.13
....relationship.others_identifier.identifier_value: 44556688
....relationship.others_identifier.__end:
....relationship.others_identifier.__begin:
....relationship.others_identifier.identifier_type: IEEE_1484.13
....relationship.others_identifier.identifier_value: 44556699
....relationship.others_identifier.__end:
```

```
....relationship.relationship_label.__begin:
....relationship.relationship.string: history_101_section_3
....relationship.relationship.locale: en-US
....relationship.relationship_label.__end:
....relationship.others_identifier.__end:
....relationship.relationship_to_them: classmate
```

7.6.2 PAPI_learner_relationship_type_type

ISO/IEC 11404 summary

```
type PAPI_learner_relationship_type_type =
enumerated // open vocabulary
(
    "classmate",
    "teacher_is",
    "teacher_of",
    "instructor_is",
    "instructor_of",
    "belongs_to",
    "belongs_with",
),
```

Description

The relationship type is an open vocabulary with the following enumerated list:

- **"classmate"**: An individual who shares learning experiences with this learner.
- **"belongs_with"**: A group that has a relationship to this learner.
- **"teacher_is"**: An individual who acts in the role of teacher to this learner.
- **"teacher_of"**: An individual who acts in the role of learner to this teacher.

7.6.3 PAPI_learner_relationship_type

ISO/IEC 11404 summary

```
type PAPI_learner_relationship_type =
record
(   // note: all components are optional; all sizes are SPM
    others_identifier_list : // Identifiers of others related to me
        arraylist(PAPI_learner_identifier_type,200),
    relations_label_list : // Name of relationship
        arraylist(mstring_type,200),
    relation_to_them_list : // My relationship to them
        arraylist(PAPI_learner_relationship_type_type,200),
    relation_to_me_list : // Their relationship to me
        arraylist(PAPI_learner_relationship_type_type,200),
),
```

Description

A single PAPI learner relationship is defined by this datatype.

The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`others_identifier_list:`** A list of identifiers of others who are related to "me".
- **`relations_label_list:`** Labels that describe the nature of the relationship.
- **`relation_to_them_list:`** How "I" am related to "them".
- **`relation_to_me_list:`** How "they" are related to "me".

7.7 PAPI learner security information datatypes

The following datatypes are used by the PAPI security personal information type.

7.7.1 PAPI_learner_security_info_type

ISO/IEC 11404 summary

```
type PAPI_learner_security_info_type =
record
(   // note: all components are optional; all sizes are SPM
    my_security_identifier_list : // database linking (key)
        arraylist(PAPI_learner_identifier_type,200),
    security_hid_list : // HID linking (cross-repository)
        arraylist(PAPI_learner_hid_type,200),
    credential_list : // Security credentials
        arraylist(PAPI_learner_security_credential_type,500),
    security_bucket :
        arraylist(PAPI_learner_bucket_type,300),
),
```

Description

PAPI learner security information is defined by this datatype.

The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`my_security_identifier_list:`** An internal database key for linking information.
- **`security_hid_list:`** An external human identifier for correlating information across data repositories.
- **`credential_list:`** A list of security credentials.
- **`security_bucket:`** A "bucket" for adding name-value pairs that provides limited extension capabilities to PAPI learner security information.

Example

The following are sample ISO/IEC 11404 (data set), XML (data instance), and DNVP (data instance) values:

```
// ISO/IEC 11404 data set
(
    my_security_identifier_list =
    (
        (
            identifier_type = "pointer",
            identifier_value = "0x12345678"
        ),
    ),
    security_hid_list =
    (
        (
            identifier_type = "IEEE_1484.13",
            identifier_value = "88990011"
        ),
    ),
    credential_list =
    (
        (
            (
                context_label= "home",
                credential_type = "password",
                credential_value = "swordfish"
            ),
            (
                context_label= "work",
                credential_type = "biometric_type_1",
                credential_value = "120398123b10931203123"
            ),
        ),
    ),
),

<!-- XML data instance ("..." is replaced by outer tags) -->
<...>
    <my_security_identifier_list>
        <security_identifier>
            <identifier_type>pointer</security_type>
            <identifier_value>0x12345678</security_value>
        </security_identifier>
    </my_security_identifer_list>
    <my_security_hid_list>
        <learner_hid>
            <identifier_type>IEEE_1484.13</identifier_type>
            <identifier_value>88990011</identifier_value>
        </learner_hid>
    </my_security_hid_list>
    <credential_list>
        <credential>
            <context_label>home</context_label>
            <credential_type>password</credential_type>
            <credential_value>swordfish</credential_value>
        </credential>
        <credential>
            <context_label>work</context_label>
            <credential_type>biometric_type_1</credential_type>
            <credential_value>120398123b10931203123</credential_value>
        </credential>
    </credential_list>
</...>

#### DNVP data instance ("..." is replaced by outer context)
....my_security_identifier.__begin:
....my_security_identifier.identifier_type: pointer
....my_security_identifier.identifier_value: 0x12345678
```

```
....my_security_identifier.__end:
....my_security_hid.__begin:
....my_security_hid.identifier_type: IEEE_1484.13
....my_security_hid.identifier_value: 88990011
....my_security_hid.__end:
....credential.__begin:
....credential.context_label: home
....credential.credential_type: password
....credential.credential_value: swordfish
....credential.__end:
....credential.__begin:
....credential.context_label: work
....credential.credential_type: biometric_type_1
....credential.credential_value: 120398123b10931203123
....credential.__end:
```

7.7.2 PAPI_learner_security_crendential_type

ISO/IEC 11404 summary

```
type PAPI_learner_security_credential_type =
record
(
    context_label :
        PAPI_learner_context_label_type,
    credential_type :
        PAPI_learner_identifier_type_type,
    credential_value :
        octetstring, // SPM: 8192
),
```

Description

An individual security credential. The meaning , namespace, scoping, and resolution of this feature shall be implementation-defined. The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`context_label:`** The context for use of this data element.
- **`credential_type:`** The type of credential.
- **`credential_value:`** The value of the credential.

7.8 PAPI learner preference information datatypes

The following datatypes are used by the PAPI preference personal information type.

7.8.1 PAPI_learner_preference_info_type

ISO/IEC 11404 summary

```
type PAPI_learner_preference_info_type =
record
(   // note: all components are optional; all sizes are SPM
    my_preference_identifier_list : // database linking (key)
        arraylist(PAPI_learner_identifier_type,200),
    preference_hid_list : // HID linking (cross-repository)
        arraylist(PAPI_learner_hid_type,200),
    pre_include_preference_hid_list : // include before this list
        arraylist(PAPI_learner_hid_type,100),
    post_include_preference_hid_list : // include after this list
        arraylist(PAPI_learner_hid_type,100),
    device_preference_list : // Device preferences
        arraylist(PAPI_learner_device_preference_type,50),
    preference_bucket :
        arraylist(PAPI_learner_bucket_type,100),
),
```

Description

PAPI learner preference information is defined by this datatype.

This data element describes preferences that allow information technology systems and learning technology systems to accommodate the needs of learners and optimize learning experiences. The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`my_preference_identifier_list:`** An internal database key for linking information.
- **`preference_hid_list:`** An external human identifier for correlating information across data repositories.
- **`pre_include_preference_hid_list:`** Preferences that are include *before* this preference set.
- **`post_include_preference_hid_list:`** Preferences that are include *after* this preference set.
- **`device_preference_list:`** Device preferences for individual types of I/O devices.
- **`preference_bucket:`** A "bucket" for adding name-value pairs that provides limited extension capabilities to PAPI learner preference information.

Example

The following are sample ISO/IEC 11404 (data set), XML (data instance), and DNVP (data instance) values:

```
// ISO/IEC 11404 data set
// Example for a deaf, but not mute person
(
    my_preference_identifier_list =
    (
        (
            identifier_type = "pointer",
            identifier_value = "0x12345678"
        ),
    ),
    preference_hid_list =
    (
        (
            identifier_type = "IEEE_1484.13",
            identifier_value = "22334455"
        ),
    ),
    pre_include_preference_hid_list =
    (
        (
            identifier_type = "IEEE_1484.13",
            identifier_value = "school_preferences"
        ),
    ),
    post_include_preference_hid_list =
    (
        (
            identifier_type = "IEEE_1484.13",
            identifier_value = "deaf_preferences"
        ),
        (
            identifier_type = "IEEE_1484.13",
            identifier_value = "not_mute_preferences"
        ),
    ),
    device_preference_list =
    (
        audio_list =
        (
            (
                output =
                (
                    context_label = "all",
                    preference_rating = -10000,
                    preference_priority = 10000,
                ),
            ),
        ),
    ),
),

<!-- XML data instance ("..." is replaced by outer tags) -->
<!-- Example for a deaf, but not mute person -->
<...>
    <my_preference_identifier_list>
        <preference_identifier>
            <identifier_type>pointer</security_type>
            <identifier_value>0x12345678</security_value>
        </preference_identifier>
    </my_preference_identifer_list>
    <my_preference_hid_list>
        <learner_hid>
            <identifier_type>IEEE_1484.13</identifier_type>
            <identifier_value>22334455</identifier_value>
        </learner_hid>
    </my_preference_hid_list>
    <pre_include_preference_hid_list>
        <learner_hid>
            <identifier_type>IEEE_1484.13</identifier_type>
            <identifier_value>school_preferences</identifier_value>
        </learner_hid>
    </pre_include_preference_hid_list>
    <post_include_preference_hid_list>
        <learner_hid>
```

```
            <identifier_type>IEEE_1484.13</identifier_type>
            <identifier_value>deaf_preferences</identifier_value>
        </learner_hid>
        <learner_hid>
            <identifier_type>IEEE_1484.13</identifier_type>
            <identifier_value>not_mute_preferences</identifier_value>
        </learner_hid>
    </post_include_preference_hid_list>
    <device_reference_list>
        <audio_list>
            <output>
                <context_label>all</context_label>
                <preference_rating>-10000</preference_rating>
                <preference_priority>10000</preference_priority>
            </output>
        </audio_list>
    </device_reference_list>
</...>

#### DNVP data instance ("..." is replaced by outer context)
#### Example for a deaf, but not mute person
....my_preference_identifier.__begin:
....my_preference_identifier.identifier_type: pointer
....my_preference_identifier.identifier_value: 0x12345678
....my_preference_identifier.__end:
....my_preference_hid.__begin:
....my_preference_hid.identifier_type: IEEE_1484.13
....my_preference_hid.identifier_value: 22334455
....my_preference_hid.__end:
....pre_include_preference.__begin:
....pre_include_preference.identifier_type: IEEE_1484.13
....pre_include_preference.identifier_value: school_preferences
....pre_include_preference.__end:
....post_include_preference.__begin:
....post_include_preference.identifier_type: IEEE_1484.13
....post_include_preference.identifier_value: deaf_preferences
....post_include_preference.__end:
....post_include_preference.__begin:
....post_include_preference.identifier_type: IEEE_1484.13
....post_include_preference.identifier_value: not_mute_preferences
....post_include_preference.__end:
....device_preference.__begin:
....device_preference.audio.__begin:
....device_preference.audio.output.context_label: all
....device_preference.audio.output.preference_rating: -10000
....device_preference.audio.output.preference_priority: 10000
....device_preference.audio.__end:
....device_preference.__end:
```

7.8.2 PAPI_learner_device_preference_type

ISO/IEC 11404 summary

```
type PAPI_learner_device_preference_type =
record
(   // note: all components are optional; all sizes are SPM
    security_list : // Security devices
        arraylist(PAPI_learner_device_io_preference_type,20),
    text_list : // Text devices
        arraylist(PAPI_learner_device_io_preference_type,20),
    speech_list : // Speech devices
        arraylist(PAPI_learner_device_io_preference_type,20),
    graphics_list : // Graphical devices
        arraylist(PAPI_learner_device_io_preference_type,20),
    audio_list : // Audio devices
        arraylist(PAPI_learner_device_io_preference_type,20),
    video_list : // Video devices
        arraylist(PAPI_learner_device_io_preference_type,20),
    tactile_list :  // Tactile devices
        arraylist(PAPI_learner_device_io_preference_type,20),
    session_choosing :  // Session choosing devices
        arraylist(PAPI_learner_device_io_preference_type,20),
    other : // Other devices
        arraylist(PAPI_learner_device_io_preference_type,100),
),
```

Description

PAPI learner device preference information is defined by this datatype.

This data element describes device preferences that allow information technology systems and learning technology systems to accommodate the needs of learners and optimize learning experiences. The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`security, text, speech, graphics, audio, video, tactile, session_choosing lists:`** Device preferences and parameters for individual types of I/O devices.
- **`other_lists:`** Other preferences that are not specified by this Standard.

7.8.3 PAPI_learner_device_io_preference

ISO/IEC 11404 summary

```
PAPI_learner_device_io_preference =
(
    input :
        PAPI_learner_device_parameter_type,
    output :
        PAPI_learner_device_parameter_type,
),
```

Description

This data element describes preferences for a pair of input/output devices. The following components define this data element. All components are optional.

- **`input:`** The corresponding input device.
- **`output:`** The corresponding output device.

7.8.4 PAPI_learner_device_parameter_type

ISO/IEC 11404 summary

```
type PAPI_learner_device_parameter_type =
(
    context : // When the device is appropriate.
        PAPI_learner_context_label_type,
    preference_name : // internationalized name
        characterstring(iso-10646-1),
    preference_rating : // Usefulness of device
        integer,
    preference_priority : // Priority among devices
        integer,
    device_name : // The name of the device
        characterstring(iso-10646-1),
    device_type : // Generic type
        characterstring(iso-10646-1),
    method : // Method of access
        characterstring(iso-10646-1),
    protocol : // Protocol stack for access
        characterstring(iso-10646-1),
    coding : // Coding method for access
        characterstring(iso-10646-1),
    encoding : // Encoding method for access
        characterstring(iso-10646-1),
    other : // Other information
        characterstring(iso-10646-1),
),
```

Description

The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`context_label:`** The context for use of this data element.
- **`preference_name:`** The internationalized (I18N) name of the preference.
- **`preference_rating:`** The usefulness of the device.
- **`preference_priority:`** The priority (importance) of this preference among similar devices.
- **`device_name:`** The machine-interpretable name of the device.
- **`device_type:`** A machine-interpretable description of device type. The specification of appropriate values is external to this Standard.
- **`method:`** The method of access. The specification of appropriate values is external to this Standard.
- **`protocol:`** The protocol used to access the device. The specification of appropriate values is external to this Standard.
- **`coding:`** The coding technique(s) used. The specification of appropriate values is external to this Standard.
- **`encoding:`** The encoding technique(s) used. The specification of appropriate values is external to this Standard.
- **`other:`** Other information. The specification of appropriate values is external to this Standard.

7.9 PAPI learner performance information datatypes

The following datatypes are used by the PAPI learner performance information type.

7.9.1 PAPI_learner_performance_info_type

ISO/IEC 11404 summary

```
type PAPI_learner_performance_info_type =
record
(   // note: all components are optional; all sizes are SPM
    my_performance_identifier_list : // database linking (key)
        arraylist(PAPI_learner_identifier_type,200),
    performance_hid_list : // HID linking (cross-repository)
        arraylist(PAPI_learner_hid_type,200),
    owner_identifier : // owner of record, not necessarily learner
        characterstring(iso-10646-1), // SPM: 1024
    recording_date_time : // when record was recorded
        time(second,10,0), // yyyymmddThhmmss
    valid_date_time_begin : // when record becomes valid
        time(second,10,0), // yyyymmddThhmmss
    valid_date_time_end : // when record expires
        time(second,10,0), // yyyymmddThhmmss
    issue_from_identifier : // who issued record
        characterstring(iso-10646-1), // SPM: 1024
    issue_date_time : // when record was issued
        time(second,10,0), // yyyymmddThhmmss
    issue_to_identifier : // who issued to, not necessarily learner
        characterstring(iso-10646-1), // SPM: 1024
    learning_experience_identifier : // experience/content ID
        characterstring(iso-10646-1), // SPM: 1024
    competency_identifier : // compentency associated with performance
        characterstring(iso-10646-1), // SPM: 1024
    granularity : // granularity of performance record
        characterstring(iso-10646-1), // SPM: 300
    performance_coding : // performance coding scheme
        characterstring(iso-10646-1), // SPM: 1024
    performance_metric : // how performance is measured
        characterstring(iso-10646-1), // SPM: 300
    peformance_value : // the value of performance for this record
        characterstring(iso-10646-1), // SPM: 2048
    certificate_list : // data certification information
        arraylist(PAPI_learner_data_certification_type,200)
    performance_bucket : // other information
        arraylist(PAPI_learner_bucket_type,100),
),
```

Description

PAPI learner performance information is defined by this datatype.

The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`my_performance_identifier_list:`** An internal database key for linking information.

- **`performance_hid_list:`** An external human identifier for correlating information across data repositories.
- **`owner_identifier:`** The learner's identifier. Note: This ID is likely to correlate to one of the learner's IDs used for other PAPI learner information.
- **`recording_date_time:`** The date-time when the record was recorded, i.e., a unique timestamp for this record.
- **`valid_date_time_begin:`** The first valid date-time of the learner performance record. Note: The performance record may only be valid for a certain period of time, e.g., its "certification" expires.
- **`valid_date_time_end:`** The first invalid (expired) date-time of the learner performance record.
- **`issue_from_identifier:`** The issuing authority of the learner performance record.
- **`issue_date_time:`** When the PAPI learner performance record was issued. This is not the same as the **`recording_date_time`** (when the record was recorded in the data repository).
- **`issue_to_identifier:`** The entity to which the learner performance record was issued. Note: This may be the learner's ID, or it may be a team/group learning ID if the team received the learner performance record.
- **`learning_experience_identifier:`** The identifier associated with the "content". Note: This might be a URL, the name of a tool, etc.. The standardization of these names is outside the scope of this Standard.
- **`granularity:`** The relative "size" of the content. This data element is a string. The meaningful values are unspecified.
- **`performance_coding_scheme:`** The type of grading, coding, measuring, etc., system in use. Note: It is not possible to standardize on a common scheme (e.g., letter grade, numeric grade), as it is expected that initially there will be many schemes. Over time, these schemes will reduce to a smaller set as the "market" demands fewer coding schemes to support more data interchange. Example: **`"US-NY-K12-LETTER-GRADE"`**, **`"ATC-610-SIMULATOR"`**.
- **`performance_metric:`** The value of permitted values.
- **`performance_value:`** The "grade", etc., that is recorded. Examples: **`"A+"`**, **`"97"`**, **`"Pass"`**.
- **`certificate_list:`** Data certification information associated with this PAPI learner performance information.
- **`performance_bucket:`** A "bucket" for adding name-value pairs that provides limited extension capabilities to PAPI learner performance information.

Example

The following are sample ISO/IEC 11404 (data set), XML (data instance), and DNVP (data instance) values:

```
// ISO/IEC 11404 data set
(
    my_performance_identifier_list =
    (
        (
            identifier_type = "pointer",
            identifier_value = "0x12345678"
        ),
    ),
    performance_hid_list =
    (
        (
            identifier_type = "IEEE_1484.13",
            identifier_value = "99887766"
        ),
    ),
    recording_date_time = 20001122T001122,
    issue_from_identifier = "registrar@xyz.edu",
    issue_date_time = 20001122T001122,
    learning_experience_identifier = "XYZ College/History 101"
    granularity = "course",
    performance_coding = "us-nys-letter-grade",
    performance_metric = "A B C D F",
    peformance_value = "A",
    certificate_list =
    (
        (
            certification_source = "certs.org",
            certification_method = "http://certs.org:1234/validate"
            certification_parameter_list = "$id $data $fields"
            certification_subset =
                "issue_date_time " +
                "issue_from_identifier " +
                "issue_to_identifier " +
                "learning_experience_identifier " +
                "performance_coding " +
                "peformance_metric " +
                "performance_value",
            certification_identifier = "123abchd1kl2jdlsjew43i5",
        )
    )
    performance_bucket =
    (
        (
            name = "time_on_task",
            value = "1h25m32s",
        ),
    ),
),

<!-- XML data instance ("..." is replaced by outer tags) -->
<...>
    <my_performance_identifier_list>
        <performance_identifier>
            <identifier_type>pointer</security_type>
            <identifier_value>0x12345678</security_value>
        </performance_identifier>
    </my_performance_identifer_list>
    <my_performance_hid_list>
        <learner_hid>
            <identifier_type>IEEE_1484.13</identifier_type>
            <identifier_value>99887766</identifier_value>
        </learner_hid>
    </my_performance_hid_list>
    <recording_date_time>20001122T001122</recording_date_time>
    <issue_from_identifier>registrar@xyz.edu</issue_from_identifier>
    <issue_date_time>20001122T001122,
    <learning_experience_identifier>
        XYZ College/History 101
    </learning_experience_identifier>
    <granularity>course</granularity>
    <performance_coding>us-nys-letter-grade</performance_coding>
    <performance_metric>A B C D F</performance_metric>
    <peformance_value>A</peformance_value>
    <certificate_list>
        <certificate>
            <certification_source>certs.org</certification_source>
            <certification_method>
                http://certs.org:1234/validate"
            <certification_method>
            <certification_parameter_list>
                $id $data $fields"
            </certification_parameter_list>
            <certification_subset>
                issue_date_time
                issue_from_identifier
                issue_to_identifier
                learning_experience_identifier
                performance_coding
                peformance_metric
                performance_value
            </certification_subset>
            <certification_identifier>
                123abchd1kl2jdlsjew43i5
            </certification_identifier>
        </certificate>
    </certificate_list>
    <performance_bucket>
        <bucket>
            <name>time_on_task</name>
            <value>1h25m32s</value>
        </bucket>
    </performance_bucket>
</...>

#### DNVP data instance ("..." is replaced by outer context)
....my_performance_identifier.__begin:
....my_performance_identifier.identifier_type: pointer
....my_performance_identifier.identifier_value: 0x12345678
....my_performance_identifier.__end:
....my_performance_hid.__begin:
....my_performance_hid.identifier_type: IEEE_1484.13
....my_performance_hid.identifier_value: 99887766
....my_performance_hid.__end:
....recording_date_time: 20001122T001122
....issue_from_identifier: registrar@xyz.edu
....issue_date_time: 20001122T001122
....learning_experience_identifier: XYZ College/History 101
....granularity: course
....performance_coding: us-nys-letter-grade
....performance_metric: A B C D F
```

```
....peformance_value: A
....certificate.__begin:
....certificate.certification_source: certs.org
....certificate.certification_method: http://certs.org:1234/validate
....certificate.certification_parameter_list: $id $data $fields
....certificate.certification_subset: issue_date_time \
    issue_from_identifier \
    issue_to_identifier \
    learning_experience_identifier \
    performance_coding \
    peformance_metric \
    performance_value
....certificate.certification_identifier: 123abchd1kl2jdlsjew43i5
....certificate.__end:
....performance_bucket.__begin:
....performance_bucket.name: time_on_task
....performance_bucket.value: 1h25m32s
....performance_bucket.__end:
```

7.10 PAPI learner portfolio information datatypes

The following datatypes are used by the PAPI learner portfolio information type.

7.10.1 PAPI_learner_portfolio_info_type

ISO/IEC 11404 summary

```
type PAPI_learner_portfolio_info_type =
record
(   // note: all components are optional; all sizes are SPM values
    my_portfolio_identifier : // database linking (key)
        arraylist(PAPI_learner_identifier_type,200),
    portfolio_hid : // HID linking (cross-repository)
        arraylist(PAPI_learner_hid_type,200),
    media_id_type : // media type
        arraylist(MIME_type,200),
    media_id : // media associated with works
        arraylist(URI_type,200),
    media_lom_list : // reference to LTSC LOM record
        arraylist(LTSC_LOM_reference_type,200),
    media_papi_learner_performance_list : // associated performance
        arraylist(LTSC_PAPI_learner_performance_reference_type,200),
    media_competency_definition_list : // associated competency
        arraylist(LTSC_competency_definition_reference_type,200),
    certificate_list : // data certification information
        arraylist(PAPI_learner_data_certification_type,200)
    portfolio_bucket : // other information
        arraylist(PAPI_learner_bucket_type,100),
),
```

Description

PAPI learner portfolio information is defined by this datatype.

The following components define this data element. All components are optional. All sizes or limits are smallest permitted maximum values.

- **`my_portfolio_identifier_list:`** An internal database key for linking information.
- **`portfolio_hid_list:`** An external human identifier for correlating information across data repositories.
- **`media_id_type:`** The media associated with the learner's accomplishments or works.
- **`media_lom_list:`** A list of LTSC Learning Object Metadata (LOM) references that relate to the learner's accomplishments or works.
- **`media_papi_learner_performance_list:`** A list of references to PAPI learner performance information records that are related to the learner's accomplishments or works.
- **`media_competency_definition_list:`** A list of references to LTSC Competency Definitions that are related to the learner's accomplishments or works.
- **`certificate_list:`** Data certification information associated with this PAPI learner performance information.
- **`portfolio_bucket:`** A "bucket" for adding name-value pairs that provides limited extension capabilities to PAPI learner portfolio information.

Example

The following are sample ISO/IEC 11404 (data set), XML (data instance), and DNVP (data instance) values:

```
// ISO/IEC 11404 data set
(
    my_portfolio_identifier_list =
    (
        (
            identifier_type = "pointer",
            identifier_value = "0x12345678"
        ),
    ),
    portfolio_hid_list =
    (
        (
            identifier_type = "IEEE_1484.13",
            identifier_value = "55443322"
        ),
    ),
    media_id_type = "video/mpeg",
    media_id = "http://mystuff.org/my_paintings/tour.mpeg",
    certificate_list =
    (
        (
            certification_source = "certs.org",
            certification_method = "http://certs.org:1234/validate"
            certification_parameter_list = "$id $data $fields"
            certification_subset =
                "media_type " +
                "media_id " +
                "media_id/* ",
            certification_identifier = "asdk12s782349238s43i5x",
        )
    )
),

<!-- XML data instance ("..." is replaced by outer tags) -->
<...>
    <my_portfolio_identifier_list>
        <portfolio_identifier>
            <identifier_type>pointer</security_type>
            <identifier_value>0x12345678</security_value>
        </portfolio_identifier>
    </my_portfolio_identifer_list>
    <my_portfolio_hid_list>
        <learner_hid>
            <identifier_type>IEEE_1484.13</identifier_type>
            <identifier_value>55443322</identifier_value>
        </learner_hid>
    </my_portfolio_hid_list>
    <media_id_type>video/mpeg</media_id_type>
    <media_id>http://mystuff.org/my_paintings/tour.mpeg</<media_id>
    <certificate_list>
        <certificate>
            <certification_source>certs.org</certification_source>
            <certification_method>
                http://certs.org:1234/validate"
            <certification_method>
            <certification_parameter_list>
                $id $data $fields"
            </certification_parameter_list>
            <certification_subset>
                media_type
                media_id
                media_id/*
            </certification_subset>
            <certification_identifier>
                asdk12s782349238s43i5x
            </certification_identifier>
        </certificate>
    </certificate_list>
</...>

#### DNVP data instance ("..." is replaced by outer context)
....my_portfolio_identifier.__begin:
....my_portfolio_identifier.identifier_type: pointer
....my_portfolio_identifier.identifier_value: 0x12345678
....my_portfolio_identifier.__end:
....my_portfolio_hid.__begin:
....my_portfolio_hid.identifier_type: IEEE_1484.13
....my_portfolio_hid.identifier_value: 99887766
....my_portfolio_hid.__end:
....certificate.__begin:
....certificate.certification_source: certs.org
....certificate.certification_method: http://certs.org:1234/validate
....certificate.certification_parameter_list: $id $data $fields
....certificate.certification_subset: media_type media_id media_id/*
....certificate.certification_identifier: asdk12s782349238s43i5x
....certificate.__end:
```

8 Bindings

PAPI learner information is bound to codings, APIs, and protocols.

Note 1: All PAPI Learner bindings (codings, APIs, protocols, etc.) are "conditionally normative":

- An implementation conformance statement (ICS) describes which features of this Standard the implementation claims conformance to.
- The ICS describes or satisfies certain conditions, as defined elsewhere in this Standard.
- Those portions of this Standard then become normative for *that* implementation.

Example: An implementation that claims it conforms to "PAPI Learner XML coding binding" is required to satisfy Annex C, XML Coding Binding, but is not required to satisfy Annex D, DNVP (Dotted Name-Value Pair) Coding Binding. If an implementation claims it conforms to both the XML and DNVP coding bindings, then it is required to satisfy both Annex C and Annex D.

Note 2: It is intended to support several coding bindings (XML, ASN.1, C, LISP), API bindings (C, C++, Java, ECMAscript, Perl, Tcl, Visual Basic, LISP), and protocol bindings (HTTP tunneling, DCTP, CORBA) in this edition or in future editions of this Standard.

9 Encodings

PAPI learner information is encoded as data formats, calling conventions, and communication layers.

Note 1: All PAPI Learner encodings are defined by their respective PAPI Learner bindings (codings, APIs, protocols, etc.).

Note 2: All PAPI Learner bindings are "conditionally normative". See Clause 8, Bindings.

Note 3: It is intended to support several data formats (ASCII, ISO/IEC 646, ISO/IEC 8859-1, ISO/IEC 10646-1), calling conventions (Unix, Linux, Win32), and communication layers (HTTP, HTTPS, IIOP, SSL, FTP, TCP, IP) in this edition or a future edition of this Standard.

10 Annex A: Bibliography (informative)

This Annex is informative and not normative.

The following are related documents:

- *** TO BE SUPPLIED ***
- *** TO BE UPDATED ***
- ISO JTC1 CAW (Cultural Adaptability Workshop)
- ISO/IEC JTC1 SC22 WG11
- ISO/IEC JTC1 SC22 WG20
- ISO/IEC JTC1 SC32 WG2
- ISO/IEC JTC1 SC35
- ISO/IEC JTC1 SC36
- NCITS L8
- NCITS T2
- IMS Profile Scope
- Specification of K-12 student records for New York State.
- University of California, Project Leap Architecture, dated 1998-01-04.

11 Annex B: ISO/IEC 11404 data model summary (informative)

This Annex is informative and not normative.

The following information is a summary of datatype definitions described elsewhere in this Standard.

Note 1: ISO/IEC 11404 notation is similar to many structured programming languages. Each data element is declared as the pair: **`"identifier : datatype"`**. A datatype may be a native ISO/IEC 11404 datatype, a generated datatype (e.g., **`"record"`**, **`"array"`**), or a created datatype (e.g., **`"type X = octetstring"`** creates a type **`"X"`**, and **`"Y : X"`** then declares **`Y`** with effective datatype **`"octetstring"`**). Statements are separated by commas, grouping is by parentheses, and datatypes may nest.

Note 2: Unless specified otherwise, all data elements are optional.

11.1 Foundational datatypes

The following is a summary of foundational datatypes in ISO/IEC 11404 notation.

```
////////////////////////////////////////////////////////////
//// Foundation datatypes for other PAPI datatypes.
////////////////////////////////////////////////////////////

// The context label associated with a data element.
// Example: "!(japan)" might mean "not within the context
// of Japan"; implementation-defined meaning
type PAPI_learner_context_label_type =
    characterstring(iso-10646-1),  // SPM: 500

// A shorthand for multiple array elements
type arraylist(type_spec,size) =
    array (0..size-1) of (type_spec),

// A multilingual string.
//     string: The localized string.
//     map: The localization (L10N) mapping, also known
//             as "locale".
type mlstring_type =
record
(
    string:
        characterstring(iso-10646-1),  // SPM: 1000
    locale:
        characterstring(iso-646),  // SPM: 255
),

// An array of multilingual strings.
// Also known as a "message cataloge" feature.
type mlstring_array_type(limit) =
    array (0..limit-1) of (mlstring_type),

// A bucket for adding name-value pairs to a PAPI Learner
// information type.  This feature gives some limited extension
// capability that works on all strictly conforming systems.
type PAPI_learner_bucket_type =
record
(
    name:
        characterstring(iso-10646-1),  // SPM: 200
    value:
        octetstring, // SPM: 4096
),

// An internal identifier that is used to link across
// databases.
type PAPI_learner_identifier_type =
record
(
    context_label :
        PAPI_learner_context_label_type,
    identifier_type :
        PAPI_learner_identifier_type_type,
    identifier_value :
        octetstring, //   // SPM: 1024
),

// An external identifier that is used to correlate
// PAPI learner information across repositories.
type PAPI_learner_hid_type =
record
(
    context_label :
        PAPI_learner_context_label_type,
    identifier_type :
        PAPI_learner_identifier_type_type,
    identifier_value :
        octetstring, //   // SPM: 1024
),

// This datatype describes the identifier type (URL, key,
// pattern, etc.).
type PAPI_learner_identifier_type_type =
    octetstring,

// This datatype describes the data certification information.
type PAPI_learner_data_certification_type =
record
(   // note: all components are optional; all sizes are SPM
    certification_source : // who certified the record
        octetstring, // SPM: 2048
    certification_method : // how it was certified
        octetstring, // SPM: 1024
    certification_parameter_list :
        octetstring, // SPM: 16384
    certification_subset : // which data elements were certified
```

```
        octetstring, // SPM: 1024
    certification_identifier : // ID associated with cert. event
        octetstring, // SPM: 2048
    certification_bucket : // other information
        arraylist(PAPI_learner_bucket_type,100),
),
```

11.2 PAPI learner personal information datatypes

The following is a summary of PAPI learner personal information datatypes in ISO/IEC 11404 notation.

```
////////////////////////////////////////////////////////////
//// PAPI learner personal information
////////////////////////////////////////////////////////////

// PAPI learner personal information type.
type PAPI_learner_personal_info_type =
record
(   // Note: All components are optional; all sizes are SPM
    my_personal_identifier_list : // database linking (key)
        arraylist(PAPI_learner_identifier_type,200),
    personal_hid_list : // HID linking (cross-repository)
        arraylist(PAPI_learner_hid_type,200),
    name_list :
        arraylist(PAPI_learner_name_type,40),
    telephone_list :
        arraylist(PAPI_learner_telephone_type,15),
    email_contact_list :
        arraylist(PAPI_learner_email_address_type,25),
    postal_address_list :
        arraylist(PAPI_learner_postal_address_type,10),
    personal_bucket :
        arraylist(PAPI_learner_bucket_type,100),
),

// The PAPI learner name.  Several name varieties are necessary
// for different application scenarios.
type PAPI_learner_name_type =
record
(   // note: all components are optional; all sizes are SPM values
    context_label : // e.g., "!(japan)"
        PAPI_learner_context_label_type,
    official_name : // e.g., { "Public" } { "Joseph" "Quincy" } - legal
        arraylist(PAPI_learner_formal_name_type,5),
    full_formal_name : // e.g., "Mr. Joseph Q. Public, PhD" name/titles
        arraylist(PAPI_learner_full_name_type,5),
    mrms_formal_name : // e.g., "Mr. Joseph Q. Public" - Mr./Ms. name
        arraylist(PAPI_learner_full_name_type,5),
    short_formal_name : // e.g., "Mr. Public" - within formal group
        arraylist(PAPI_learner_full_name_type,5),
    full_name : // e.g., "Joseph Q. Public" - in print
        arraylist(PAPI_learner_full_name_type,5),
    sort_name : // e.g., "Public, Joseph Q."
        arraylist(PAPI_learner_full_name_type,5),
    full_informal_name : // e.g., "Joe Public" - within a group
```

```
        arraylist(PAPI_learner_full_name_type,5),
    short_informal_name : // e.g., "Joe" - directly to person
        arraylist(PAPI_learner_full_name_type,5),
}

// A formal name is defined as a "primary name" and a
// "secondary name".  The terms "primary" and "secondary" come
// from ICAO, IATA, UN, and ATA standards for automated
// passport information.
type PAPI_learner_formal_name_type =
(
    primary :  // Family name(s).
        characterstring(iso-10646-1), // SPM: 70
    secondary : // Given name(s).
        characterstring(iso-10646-1), // SPM: 70
),

// A localized name representation, e.g., "Mr. Smith" and
// "Smith-san".
type PAPI_learner_full_name_type =
    characterstring(iso-10646-1), // SPM: 140

// A telephone number.
type PAPI_learner_telephone_type =
record
(
    context_label :
        PAPI_learner_context_label_type,
    identifier_type :
        PAPI_learner_identifier_type_type,
    phone_number :
        octetstring, // SPM: 50
),

// An E-mail address.
type PAPI_learner_email_contact_type =
record
(
    context_label :
        PAPI_learner_context_label_type,
    email_address_type :
        PAPI_learner_identifier_type_type,
    email_address :
        octetstring, // SPM: 255
),

// Postal addressing based on
// ISO 11180, Postal Addressing
type PAPI_learner_postal_address_type =
record
(   // note: all components are optional; all sizes are SPM
    context_label :
        PAPI_learner_context_label_type,

    // The addressee.  Note: An application should harmonize
    // the values of postal addressee name, title, etc., with
    // learner name components.
    addressee_title :
        characterstring(iso-10646-1), // SPM: 35
    addressee_name_given :
        characterstring(iso-10646-1), // SPM: 70
    addressee_name_family :
        characterstring(iso-10646-1), // SPM: 70
    addressee_name_suffix :
        characterstring(iso-10646-1), // SPM: 35
    addressee_occupation :
        characterstring(iso-10646-1), // SPM: 70
    addressee_function :
        characterstring(iso-10646-1), // SPM: 70
    addressee_care_of_address :
        characterstring(iso-10646-1), // SPM: 70

    // The organization.
    organization_name :
        characterstring(iso-10646-1), // SPM: 70
    organization_activity :
        characterstring(iso-10646-1), // SPM: 70
    organization_division :
        characterstring(iso-10646-1), // SPM: 70

    // The delivery address.
    delivery_street_type :
        characterstring(iso-10646-1), // SPM: 35
    delivery_street_name :
        characterstring(iso-10646-1), // SPM: 70
    delivery_street_id_number :
        characterstring(iso-10646-1), // SPM: 20
    delivery_supplementary_address :
        characterstring(iso-10646-1), // SPM: 70
    delivery_city :
        characterstring(iso-10646-1), // SPM: 35
    delivery_po_box :
        characterstring(iso-10646-1), // SPM: 20
    delivery_postcode :
        characterstring(iso-10646-1), // SPM: 20
    delivery_routing :
        characterstring(iso-10646-1), // SPM: 20
    delivery_office :
        characterstring(iso-10646-1), // SPM: 35
    delivery_territory :
        characterstring(iso-10646-1), // SPM: 35
    delivery_country :
        characterstring(iso-10646-1), // SPM: 35
),
```

11.3 PAPI learner relations information datatypes

The following is a summary of PAPI learner relations information datatypes in ISO/IEC 11404 notation.

```
////////////////////////////////////////////////////////////
//// PAPI learner relations information
////////////////////////////////////////////////////////////
```

```
// PAPI learner relations information type.
type PAPI_learner_relations_info_type =
record
(   // note: all components are optional; all sizes are SPM
    my_relations_identifier_list : // database linking (key)
        arraylist(PAPI_learner_identifier_type,200),
    relations_hid_list : // HID linking (cross-repository)
        arraylist(PAPI_learner_hid_type,200),
    relationship_list : // List of individual relationships
        arraylist(PAPI_learner_relationship_type,200),
    relations_bucket : // Other information
        arraylist(PAPI_learner_bucket_type,200),
),

// PAPI learner relationship type.
type PAPI_learner_relationship_type =
record
(   // note: all components are optional; all sizes are SPM
    others_identifier_list : // Identifiers of others related to me
        arraylist(PAPI_learner_identifier_type,200),
    relations_label_list : // Name of relationship
        arraylist(mstring_type,200),
    relation_to_them_list : // My relationship to them
        arraylist(PAPI_learner_relationship_type_type,200),
    relation_to_me_list : // Their relationship to me
        arraylist(PAPI_learner_relationship_type_type,200),
),

// The datatype for types of relationships.
type PAPI_learner_relationship_type_type =
enumerated // open vocabulary
(
    "classmate",
    "teacher_is",
    "teacher_of",
    "belongs_to",
    "belongs_with",
),
```

11.4 PAPI learner security information datatypes

The following is a summary of PAPI learner security information datatypes in ISO/IEC 11404 notation.

```
////////////////////////////////////////////////////////////
//// PAPI learner security information
////////////////////////////////////////////////////////////

// PAPI learner security information type.
type PAPI_learner_security_info_type =
record
(   // note: all components are optional; all sizes are SPM
    my_security_identifier_list : // database linking (key)
        arraylist(PAPI_learner_identifier_type,200),
    security_hid_list : // HID linking (cross-repository)
        arraylist(PAPI_learner_hid_type,200),
    credential_list : // Security credentials
        arraylist(PAPI_learner_security_credential_type,500),
    security_bucket :
        arraylist(PAPI_learner_bucket_type,300),
),

// A security credential.
type PAPI_learner_security_credential_type =
record
(
    context_label :
        PAPI_learner_context_label_type,
    credential_type :
        PAPI_learner_identifier_type_type,
    credential_value :
        octetstring, // SPM: 8192
),
```

11.5 PAPI learner preference information datatypes

The following is a summary of PAPI learner preference information datatypes in ISO/IEC 11404 notation.

```
////////////////////////////////////////////////////////////
//// PAPI learner preference information
////////////////////////////////////////////////////////////

// PAPI learner preference information.
type PAPI_learner_preference_info_type =
record
(   // note: all components are optional; all sizes are SPM
    my_preference_identifier_list : // database linking (key)
        arraylist(PAPI_learner_identifier_type,200),
    preference_hid_list : // HID linking (cross-repository)
        arraylist(PAPI_learner_hid_type,200),
    pre_include_preference_hid_list : // include before this list
        arraylist(PAPI_learner_hid_type,100),
    post_include_preference_hid_list : // include after this list
        arraylist(PAPI_learner_hid_type,100),
    device_preference_list : // Device preferences
        arraylist(PAPI_learner_device_preference_type,50),
    preference_bucket :
        arraylist(PAPI_learner_bucket_type,100),
),

// PAPI learner device preferences.
type PAPI_learner_device_preference_type =
record
(   // note: all components are optional; all sizes are SPM
    security_list : // Security devices
        arraylist(PAPI_learner_device_io_preference_type,20),
    text_list : // Text devices
        arraylist(PAPI_learner_device_io_preference_type,20),
    speech_list : // Speech devices
        arraylist(PAPI_learner_device_io_preference_type,20),
    graphics_list : // Graphical devices
        arraylist(PAPI_learner_device_io_preference_type,20),
    audio_list : // Audio devices
        arraylist(PAPI_learner_device_io_preference_type,20),
    video_list : // Video devices
        arraylist(PAPI_learner_device_io_preference_type,20),
    tactile_list :  // Tactile devices
        arraylist(PAPI_learner_device_io_preference_type,20),
    session_choosing :  // Session choosing devices
        arraylist(PAPI_learner_device_io_preference_type,20),
    other : // Other devices
        arraylist(PAPI_learner_device_io_preference_type,100),
),

// Preferences for a pair of input/output devices.
PAPI_learner_device_io_preference =
(
    input :
        PAPI_learner_device_parameter_type,
    output :
        PAPI_learner_device_parameter_type,
),

// A set of device preferences for an individual device.
type PAPI_learner_device_parameter_type =
(
    context : // When the device is appropriate.
        PAPI_learner_context_label_type,
    preference_name : // internationalized name
        characterstring(iso-10646-1),
    preference_rating : // Usefulness of device
        integer,
    preference_priority : // Priority among devices
        integer,
    device_name : // The name of the device
        characterstring(iso-10646-1),
    device_type : // Generic type
        characterstring(iso-10646-1),
    method : // Method of access
        characterstring(iso-10646-1),
    protocol : // Protocol stack for access
        characterstring(iso-10646-1),
    coding : // Coding method for access
        characterstring(iso-10646-1),
    encoding : // Encoding method for access
        characterstring(iso-10646-1),
    other : // Other information
        characterstring(iso-10646-1),
),
```

11.6 PAPI learner performance information datatypes

The following is a summary of PAPI learner performance information datatypes in ISO/IEC 11404 notation.

```
////////////////////////////////////////////////////////////
//// PAPI learner performance information
////////////////////////////////////////////////////////////

//
// PAPI learner performance information.
//
type PAPI_learner_performance_info_type =
record
(   // note: all components are optional; all sizes are SPM
    my_performance_identifier_list : // database linking (key)
        arraylist(PAPI_learner_identifier_type,200),
    performance_hid_list : // HID linking (cross-repository)
        arraylist(PAPI_learner_hid_type,200),
    owner_identifier : // owner of record, not necessarily learner
        characterstring(iso-10646-1), // SPM: 1024
    recording_date_time : // when record was recorded
        time(second,10,0), // yyyymmddThhmmss
    valid_date_time_begin : // when record becomes valid
        time(second,10,0), // yyyymmddThhmmss
    valid_date_time_end : // when record expires
        time(second,10,0), // yyyymmddThhmmss
    issue_from_identifier : // who issued record
        characterstring(iso-10646-1), // SPM: 1024
    issue_date_time : // when record was issued
        time(second,10,0), // yyyymmddThhmmss
    issue_to_identifier : // who issued to, not necessarily learner
        characterstring(iso-10646-1), // SPM: 1024
    learning_experience_identifier : // experience/content ID
        characterstring(iso-10646-1), // SPM: 1024
    competency_identifier : // compentency associated with performance
        characterstring(iso-10646-1), // SPM: 1024
    granularity : // granularity of performance record
        characterstring(iso-10646-1), // SPM: 300
    performance_coding : // performance coding scheme
        characterstring(iso-10646-1), // SPM: 1024
    performance_metric : // how performance is measured
        characterstring(iso-10646-1), // SPM: 300
    peformance_value : // the value of performance for this record
        characterstring(iso-10646-1), // SPM: 2048
    certificate_list : // data certification information
        arraylist(PAPI_learner_data_certification_type,200)
    performance_bucket : // other information
        arraylist(PAPI_learner_bucket_type,100),
),
```

11.7 PAPI learner portfolio information datatypes

The following is a summary of PAPI learner portfolio information datatypes in ISO/IEC 11404 notation.

```
////////////////////////////////////////////////////////////
//// PAPI learner portfolio information
////////////////////////////////////////////////////////////
```

```
// PAPI learner portfolio information.
type PAPI_learner_portfolio_info_type =
record
(   // note: all components are optional; all sizes are SPM values
    my_portfolio_identifier : // database linking (key)
        arraylist(PAPI_learner_identifier_type,200),
    portfolio_hid : // HID linking (cross-repository)
        arraylist(PAPI_learner_hid_type,200),
    media_id_type : // media type
        arraylist(MIME_type,200),
    media_id : // media associated with works
        arraylist(URI_type,200),
    media_lom_list : // reference to LTSC LOM record
        arraylist(LTSC_LOM_reference_type,200),
    media_papi_learner_performance_list : // associated performance
        arraylist(LTSC_PAPI_learner_performance_reference_type,200),
    media_competency_definition_list : // associated competency
        arraylist(LTSC_competency_definition_reference_type,200),
    certificate_list : // data certification information
        arraylist(PAPI_learner_data_certification_type,200)
    portfolio_bucket : // other information
        arraylist(PAPI_learner_bucket_type,100),
),
```

12 Annex C: XML coding binding (conditionally normative)

If a PAPI Learner application includes the statement "PAPI Learner XML coding *[implementation variety]*" in its implementation conformance statement, then that application shall conform to the requirements of this Annex.

Note: The implementation varieties are defined in Clause 4, Conformance, and summarized in subclause 4.2, Conformance Labels.

12.1 Generating and producing XML

The following rules describe the transformation of PAPI Learner data elements, as described by this Standard and by ISO/IEC 11404 notation, to XML records.

- **Rule 1:** For each data element in ISO/IEC 11404 notation, map all identifiers to XML tags, except as noted in Rule 2 below. Balanced XML tags delimit the boundary of the value associated with the data element. The nesting of the XML tags represents the structure of data elements, as described by its "aggregate datatype generator" (ISO/IEC 11404 terminology). For array and sequence aggregates, (1) an XML tag of the same name as the identifier of the aggregate represents the group of aggregates, (2) the individual data elements are represented by repeated XML tags based on the identifier of the aggregate minus the suffix **`"_list"`** or **`"_bucket"`**, not the index of the element.
- **Rule 2:** Map all **`mlstring_type`** datatypes to:
    - **Rule 2A:** The **`locale`** element of **`mlstring_type`** sets the **`LANG`** attribute in parent XML element.
    - **Rule 2B:** The **`string`** element sets content of parent tag (i.e., the current target).

  Map all PAPI learner personal information to ...[editor's note: "vCard" mapping to be added here]
- **Rule 3:** Transform the following XML tags (wildcard notation):
  `PAPI_learner_*`
  to the following XML tags (wildcard notation):
  `IEEE_LTSC_PAPI_learner_*`

All data produced shall be well-formed XML.

Rationale

The following is a rationale for these three rules for <u>*this specific*</u> transformation of the PAPI Learner datatypes.

Note: This XML binding (PAPI Learner $\rightarrow$ XML) requires 3 transformation rules. ***Other standards and different XML bindings may require more, fewer, or different transformation rules.***

Rationale for Rule 1

Rule 1 is the main transformation from ISO/IEC 11404 datatypes to XML tagging conventions. The following examples use the following definition to illustrate the transformations:

```
A: record
(
    B: integer,
    C: record
    (
        D: integer,
        E: characterstring(iso-10646-1),
    ),
    F_list: array (0..limit) of (integer),
    G: sample_mlstring_list_type,
)
```

The first sentence, "for each data element in ISO/IEC 11404 notation, map all identifiers to XML tags", transforms identifiers, e.g., "**`X:`**" Þ "**`<X>`**".

The second sentence, "balanced XML tags delimit the boundary of the value associated with the data element", requires that (1) the tags are balanced, and (2) the value of the data element is between the tags, e.g., "**`X: 17`**" Þ "**`<X>17</X>`**".

The third sentence, "the nesting of the XML tags represents the structure of data elements, as described by its aggregate datatype generator", requires that the nesting implied in aggregates (records, arrays, sequences/lists) results in similar nesting of the XML tags. Using the definition of **`A`** above, the following nesting is implied for elements **`B`**, **`C`**, **`D`**, and **`E`**:

```
<A>
    <B>...</B>
    <C>
        <D>...</D>
        <E>...</E>
    </C>
    ...
</A>
```

The fourth sentence, "for array and sequence aggregates, data elements are represented by repeated XML tags based on the identifier of the aggregate, not the index of the element", requires arrays and sequences (lists) to be represented as multiple tags with the same name — a typical XML style convention. For example, the data element **`F`** would be represented as:

```
<!-- correct XML binding of F_list -->
<A>
    ...
    <F_list>
        <F>...</F>
        <F>...</F>
        <F>...</F>
    </F_list>
    ...
</A>
```

but not as:

```
<!-- incorrect XML binding of F_list -->
<A>
```

```
        ...
        <F_list>
            <0>...</0>
            <1>...</1>
            <2>...</2>
        </F_list>
        ...
    </A>
```

Rationale for Rule 2

PAPI Learner records use several specialized datatypes, such as multilingual datatypes for describing certain `characterstring`-type data elements that must be represented in a multilingual and multicultural context — commonly called internationalization (I18N) and localization (L10N) features. Below, is a sample version of a multilingual data type that is not intended to clash with definitions of other multilingual datatypes defined elsewhere in this Standard. In this illustration, the datatype **`sample_mlstring_type`** represents a single pair: a localized string and a locale specification (L10N mapping). The datatype **`sample_mlstring_array_type`** represents an array of these string pairs. In this example, the array **`example_remarks`** contains three elements, each element is a pair of strings. Presumably, an application would choose the appropriate string from use **`example_remarks`** based on the country (locale) that the application was operating in. The following are sample type definitions and value definitions.

```
type sample_mlstring_type =
record
(
    L10N_string: characterstring(iso-10646-1),
    L10N_locale: string_type,
),

type sample_mlstring_array_type =
    array (0..limit) of (sample_mlstring_type),

value example_remarks:
    sample_mlstring_array_type =
(
    (
        L10N_string: "abc abc abc",
        L10N_locale: "en-US",
    ),
    (
        L10N_string: "def def def",
        L10N_map: "fr-CA",
    ),
    (
        L10N_string: "ghi ghi ghi",
        L10N_map: "de-DE",
    ),
),
```

Rule 2, along with array handling of Rule 1, transforms these data elements into the following XML:

```
<example_remarks LANG="en-US">abc abc abc</example_remarks>
<example_remarks LANG="fr-CA">def def def</example_remarks>
```

```
    <example_remarks LANG="de-DE">ghi ghi ghi</example_remarks>
```

[Editor's Note: Rule 2, will include "vCard" mapping which transforms PAPI learner personal information to/from "vCard" structures.]

Rationale for Rule 3

This rule is used for rewriting tags to use certain namespace conventions. This rule could have specified XML namespaces by choosing a different namespace convention (prefixes).

After Rule 3, the implementation is required to assure that the result of these transformations is well-formed XML.

12.2 Consuming and interpreting XML

The following rules describe the transformation of XML records to PAPI Learner data elements, as described by this Standard and by ISO/IEC 11404 notation.

All data consumed shall be well-formed XML.

- **Rule 1:** Transform the following XML tags (wildcard notation):
  `IEEE_LTSC_PAPI_learner_*`
  to the following XML tags (wildcard notation):
  `PAPI_learner_*`
- **Rule 2:** Transform the following:
  - **Rule 2A:** The `LANG` attribute of the XML element sets the `locale` element of the corresponding `mlstring_type` data element.
  - **Rule 2B:** The contents of the tagged element sets the `string` element of the corresponding `mlstring_type` data element.
- **Rule 3:** For each XML tag, that is associated with an identifier defined by a PAPI Learner data element in this Standard, its corresponding opening and closing balanced XML tags are matched. For each XML tag, except as modified in Rule 2 above, map each XML tag to the corresponding data element identifier. The nesting of the XML tags represents the nesting of the data elements, i.e., the reverse of the operation in Rule #1 of subclause 12.1, Generating and Interpreting XML, above. The contents of each tagged element is converted to the value of the corresponding data element.

Rationale

Rationale for Rule 1

Before processing, the implementation is assured that it is consuming and interpreting well-formed XML.

This rule strips the XML namespace prefixes and suffixes as necessary. In this illustration, XML namespaces were not used, but a namespace prefix ("`IEEE_LTSC_`") was used to reduce the possibility of namespace collisions.

Rationale for Rule 2

This rule does the reverse mapping from the **`LANG`** attribute to the **`mlstring_type`** datatype. This rule is careful to transform only known **`mlstring_type`** data elements because all other XML **`LANG`** attributes do not correspond to **`mlstring_type`** data elements in this Standard.

[Editor's Note: In the next draft, Rule 2, will include "vCard" mapping which transforms PAPI learner personal information to/from "vCard" structures.]

Rationale for Rule 3

This rule handles the main transformation of XML tags and their contents to data elements.

The first sentence, "for each XML tag, that is associated with an identifier defined by a PAPI Learner data element in this Standard, its corresponding opening and closing balanced XML tags are matched", (1) ignores all identifiers that are unknown to this Standard, and (2) properly pairs them.

The second sentence, "for each XML tag, except as modified in Rule 2 above, map each XML tag to the corresponding data element identifier", creates the association with data elements, but does not assign the values of the data elements..

The third sentence, "the nesting of the XML tags represents the nesting of the data elements, i.e., the reverse of the operation in Rule #1 of subclause 19.1", assures that the internal structure of the XML tags, to the extent required by this Standard, agree with the internal structure of the data elements.

The fourth sentence, "the contents of each tagged element is converted to the value of the corresponding data element", transforms the contents within the XML tags to values of the data elements, i.e., it "populates" the data elements.

12.3 Representation of basic data types

The following subclauses describe the transformation of data element values to/from character representations for information interchange for use within an XML binding.

12.3.1 Characters and character strings

Data elements that are of type **`character`** shall be represented as per the XML specification.

Note 1: Special characters, such as **`"&" "<" ">" ";"`** require translation methods and, possibly, lossless translation.

Note 2: Some encodings, such as ISO-8859-1 and UTF-8 permit the direct encoding of the representation of characters such as "**©**" (the copyright symbol). Other encodings, such as ASCII, require encoding extensions, such as "**`&#169;`**", to represent these symbols.

12.3.2 Integers

Data elements that are of type **`integer`** shall be represented as per ISO/IEC 9899:1999, C Programming Language, subclause 6.4.4.1, Integer Constants; excluding "**`U`**", "**`L`**", and "**`LL`**" suffixes and their lowercase variants; and may include an optional leading sign, either plus ("**`+`**") or minus ("**`-`**"), but not both.

Examples:

```
0       // zero
23      // twenty-three
0x17    // same in hexadecimal
027     // same in octal
-34     // negative thirty-four
+34     // postitive thirty-four
+34     // same
```

12.3.3 Real numbers

Data elements that are of type **`real`**:

- if integral, may be represented integers, as specified above in subclause 19.5, Integers;
- if not integral or not represented as integers, shall be represented as per ISO/IEC 9899:1999, C Programming Language, subclause 6.4.4.2, Floating Constants; excluding "**`F`**" and "**`L`**" suffixes and their lowercase variants; and may include an optional leading sign, either plus ("**`+`**") or minus ("**`-`**"), but not both.

Examples:

```
0       // zero
0.0     // same
130.0   // one hundred thirty
1.3E2   // same
+1.3E2  // same
```

12.3.4 Date and time values

Data elements of type **`time`** shall be represented as per ISO 8601, Data elements and interchange formats — Information interchange — Representation of dates and times.

Note 1: ISO 8601 is intended to represent dates and times between 1 January 0001 and 31 December 9999 using the Gregorian calendar. It is well understood that there are anomalies with this approach, e.g., the month of September 1752 has less than 30 days, the lack of correlation of timezones prior to the introduction of transcontinental railroads, changing the beginning month of the calendar, the inability to represent dates Before the Common Era, and the inability to represent dates after 31 December 9999.

ISO 8601 is applied as follows:

- Only the basic format shall be used. Example: **`"19990102"`** and **`"030405"`** are valid, but **`"1999-01-02"`** and **`"03:04:05"`** are not. Rationale: The additional processing for extended formats may be more error prone and can reduce interoperability;

the additional lexical elements of the extended formats may interfere or conflict with other lexical, embedded, or surrounding information processing environments.

- Decimal fractions shall use the full stop character (**`"."`**), also known as the period character. Example: **`"199901020304.1"`** represents 03:04:06 AM on 2 January 1999. Note: ISO 8601 permits the use of the comma (**`","`**) and full stop characters, and specifies that comma is the preferred sign. This Standard only permits the full stop (period) character. Rationale: The comma character may interfere or conflict with other lexical, embedded, or surrounding information processing environments.

For points in time, ISO 8601 is applied as follows:

- Dates shall be represented by calendar date format only. Note: Calendar dates are represented by year, month, and date. Other formats permitted by ISO 8601, but prohibited in this Standard include ordinal date (i.e., the date is represented by three decimal digits, e.g., **`"1985032"`** is 1 February 1985) and calendar week and day number (i.e., the date is represented by the week number and the day of the week, e.g., Sunday, 1 January 1995 is **`"1994W527"`**, and Tuesday, 31 December 1996 is **`"1997W012"`**). Rationale: The additional processing for these other date formats may be more error prone and can reduce interoperability.
- Dates and times shall be represented as an ISO 8601 complete representation and shall not use the **`"T"`** time indicator. Example: **`"19990102030405"`** represents 03:04:05 AM on 2 January 1999. Rationale: The **`"T"`** time indicator is omitted because it is not necessary; ISO 8601 permits this omission when there is no ambiguity. The ambiguity is avoided by (1) permitting only ISO 8601 basic formats, and (2) requiring the identification of missing date and time components.
- The underscore character (**`"_"`**) shall be used to indicate missing components. One underscore character shall be used for each digit that is missing from the integer portion of the date or time. Example: **`"____0102"`** represents 2 January of the current year. Note: Data that has missing components (e.g., the century) can cause significant interoperability problems, such as the "Year 2000 Problem". It is recommended that implementations avoid generating data with missing components, but it may not be possible to resolve all circumstances. Rationale: ISO 8601 uses hyphens for missing components, but hyphens may interfere or conflict with other lexical, embedded, or surrounding information processing environments. ISO 8601 uses the hyphen to indicate that one component is missing (e.g., **`"--0102"`**), while this Standard uses one underscore character per digit (e.g., **`"____0102"`**), which is less error prone for information processing.
- Times shall be represented in local time or Coordinated Universal Time (UTC). Times represented in local time shall use no suffix and shall not indicate the difference between local time and UTC. Times represented in UTC shall use the "Z" suffix, as per ISO 8601. Example: **`"19990102030405"`** local time in New York City (5 hours west of UTC) is the same time as **`"19990102030905Z"`**. Rationale: The use of time zone information, other than UTC, causes additional processing for data interpretation, which may be more error prone. The prohibition of time differences (e.g., **`"19990102030405+0700"`**) eliminates the use of the plus (**`"+"`**) and minus (**`"-"`**) characters which may interfere or conflict with other lexical, embedded, or surrounding information processing environments.

For durations of time, ISO 8601 is applied as follows:

- The time duration shall begin with the prefix **`"P"`**, as per ISO 8601. Example: **`"P2Y"`** means two years. Note: Not all date and time components are required for a duration of time.
- The time duration shall use the case-sensitive designators: **`"Y"`** for years, **`"M"`** for months, **`"D"`** for days, **`"W"`** for weeks, **`"h"`** for hours, **`"m"`** for minutes, and **`"s"`** for seconds. Example: **`"P2Y10M25h2m5s"`** represents the duration 2 years, 10 months, 25 hours, 2 minutes, and 5 seconds. For durations of time, a date and time component may exceed the maximum number of units in a corresponding component of a point in time, e.g., months may exceed 12, hours may exceed 24, and minutes may exceed 60. Rationale: The use of case-sensitive designators simplifies the processing of date and time information.

Note 2: The characters described in ISO 8601 only specify the names of characters, not their encodings. This is emphasized by ISO 8601, subclause 4.4, Characters used in the representations: "The representations specified in this International Standard use digits, alphabetic characters, and special characters specified in ISO 646. ... Note 2: Encoding of characters for the interchange of dates and times is not in the scope of this Standard."

12.3.5 Void types

A void type shall have no representation and shall have no encoding.

Example: The following record

```
A: record
(
    B: integer,
    C: void,
    D: characterstring(iso-10646-1),
)
```

is represented in XML as:

```
<!-- correct XML representation of C -->
<A>
    <B>17</B>
    <D>hello</D>
</A>
```

but not as:

```
<!-- incorrect XML representation of C -->
<A>
    <B>17</B>
    <C></C>
    <D>hello</D>
</A>
```

12.4 Encoding of character representations

The encoding of character representations to octet values is specified by the XML encoding technique.

Conforming PAPI Learner data instances with XML coding binding shall be encoded in one of: ASCII, ISO/IEC 8859-1, ISO/IEC 10646-1 UTF-8, or ISO/IEC 10646-1 UTF-16.

Conforming PAPI Learner applications with XML coding binding shall support all of the following encodings: ASCII, ISO/IEC 8859-1, ISO/IEC 10646-1 UTF-8, and ISO/IEC 10646-1 UTF-16.

12.5 Handling exceptions and extensions

12.5.1 Implementation-defined behavior

The following are implementation-defined behaviors in addition to those described elsewhere in this Standard.

The following are implementation-defined behaviors in the production and consumption of XML codings:

- The maximum size, in octets, of a strictly conforming PAPI Learner data instance, as coded in XML, that may be processed successfully.
- The maximum nesting depth of XML records.
- The time zone information for data elements of **`time`** type that have unspecified timezones.

12.5.2 Unspecified behavior

The following are unspecified behaviors in addition to those described elsewhere in this Standard.

The following is unspecified behavior in the generation or interpretation of XML codings:

- The order of processing data elements.

The following is unspecified behavior in the production or consumption of XML codings:

- The use of additional whitespace characters outside those of the data element value and those required by the XML specification.

12.5.3 Undefined behavior

The following are undefined behaviors in addition to those described elsewhere in this Standard.

The following are undefined behaviors in the production or consumption of XML codings:

- The use of XML tags that correspond to extended data elements.
- The use of XML tags that correspond to reserved data elements.
- The use of XML tags or attributes not specified in this XML coding binding.
- The use of characters outside the repertoire described in this Standard.

13 Annex D: DNVP coding binding (conditionally normative)

If a PAPI Learner application includes a "PAPI Learner DNVP coding *[implementation variety]* " in its implementation conformance statement, then that application shall conform to the requirements of this Annex.

Note 1: The implementation varieties are defined in Clause 4, Conformance, and summarized in subclause 4.2, Conformance Labels.

Note 2: The Dotted Name-Value Pair (DNVP) notation is based on an RFC 822 style of messaging. RFC 822, Standard for the Format of ARPA Internet Text Messages, describes text messages that are commonly referred to as internet E-mail messages. In this style of messaging, the beginning of a message contains a header with header elements. For example,

```
From: sender@host.com
To: user@host.com
Subect: a subject line
```

might represent three header elements — a portion of a header. This kind of messaging is described in RFC 822, subclause 3.1, General Description. Additionally, RFC 2068, Hypertext Transfer Protocol — HTTP/1.1, subclause 4.2, Message Headers, itself is harmonized with RFC 822 (from RFC 2068, subclause 4.2: "HTTP header fields ... follow the same generic format as that given in Section 3.1 of RFC 822").

This style of binding (RFC-822-like) is in common use in many interchange environments, such as E-mail systems and web servers.

The following is an example of this generalized format:

```
name_1: value_1
name_2: value_2
name_3: value_3
```

13.1 Dotted Name-Value Pairs (DNVPs)

Note: The following wording was extracted, excerpted and adapted from, and harmonized with RFC 2068 (HTTP/1.1).

13.1.1 Basic lexical elements

A newline character (CRLF) is defined by its encoding.

Note 1: A newline character might be line feed (e.g., Unix/Linux), carriage return (e.g., Macintosh), or carriage return line feed combination (e.g., Windows).

For maximum interoperability, implementations should use the carriage return line feed combination for the newline character when producing data. Implementations that use only one character, either line feed or carriage return, for the newline character should ignore the other character, carriage return or line feed, respectively, when consuming data.

Leading whitespace (LWS) is defined as at least one or more space or tab characters that follow a newline.

Note 2: LWS does not include the prior newline. A sequence of space or tab characters without a prior newline is not LWS.

A control character (CTL) is a character in the range 0-31 (decimal) or the octet 127 (decimal) but not the octet 9 (HT).

A text characters is any character except control characters.

The following are token special characters:

```
(   )   <   >   @
,   ;   :   \   "
/   [   ]   ?   =
{   }   SP  HT
```

A token is one or more characters that exclude control characters or token special characters.

A string of text is consumed as a single token (and token special characters are not special) if it is quoted using double-quote marks.

Example 1: The characters

```
"abcd -(), :[]"
```

are consumed as a single token.

The backslash character ("\") may be used for single-character quoting, i.e., removing the special nature of a token special character, one character at a time.

Example 2: The characters

```
they're
```

are consumed as a single token and the single quote character (`'`) has no special meaning.

Example 3: The characters

```
"say \"hello\""
```

are consumed as a single token, the token includes the double quote characters before and after the work hello, and the quote characters that surround the word hello have no special meaning.

13.1.2 Field name and field value

A Name-Value Pair (NVP) shall consist of a field name, followed by a colon (":"), followed by its field value. A field name is a token. Field names shall be case-sensitive.

Note: The case sensitivity of this DNVP binding differs from RFC 822.

A field value may be preceded by any amount of leading white space (LWS). Conforming implementations should use a single space (SP) as LWS. A field value is zero more characters consisting of combinations of tokens and/or token special characters.

A field value of zero characters represents an empty value being associated with the field name.

Note: A Name-Value Pair starts at the beginning of a line.

13.1.3 Newline processing

A Name-Value Pair may be extended over multiple lines by preceding each extra line with at least one space (SP) or horizontal tab (HT). All linear white space, including folding, has the same semantics as a single SP character.

A newline (CRLF) shall follow a completed Name-Value Pair.

13.1.4 Syntax summary

This subclause is informative and not normative.

Note: The following BNF syntax summary is extracted, excerpted and adapted from, and harmonized with RFC 2068 (HTTP/1.1):

```
CRLF           = <newline character>
CTL            = <any US-ASCII control character
                 (characters 0 - 31) and DEL (127), except HT
(9)>
LWS            = [CRLF] 1*( SP | HT )
token          = 1*<any character except CTLs or tspecials>
tspecials      = "(" | ")" | "<" | ">" | "@"
               | "," | ";" | ":" | "\" | <">
               | "/" | "[" | "]" | "?" | "="
               | "{" | "}" | SP | HT
quoted-pair    = "\" character
quoted-string  = <">*<any char except non-quoted
                     double quotes><">
message-header = field-name ":" [ field-value ] CRLF
field-name     = token
field-value    = *( field-content | LWS )
field-content  = <the characters making up the field-value
                 and consisting of combinations of
                 tokens or tspecials>
```

13.2 Generating and producing dotted name-value pairs

The following rules describe the transformation of PAPI Learner data elements, as described by this Standard and by ISO/EC 11404 notation, to DNVP notation.

- **Rule 1:** For each data element, map all identifiers to fully-qualified field names. A fully-qualified name represents the nested structure of the data element, as described by its "aggregate datatype generator" (ISO/IEC 11404 terminology). A period (`"."`) shall separate each level of nesting in a fully-qualified field name. For array and sequence aggregates, (1) the individual data elements are represented by repeated DNVP field names based on the identifier of the aggregate minus the suffix **`"_list"`** or **`"_bucket"`**, not the index of the element, (2) each element of the array or sequence aggregate shall bracketed by two null-value DNVPs, one at the beginning of each element with the field name suffix **`".__begin"`** and one at the end of each element with the field name suffix **`".__end"`**. A field name is followed by a colon (":"), followed by the value of its associated data element.
- **Rule 2:** Transform the following DNVP field names (wildcard notation):
  ```
  PAPI_learner_*
  ```
  to the following DNVP field names (wildcard notation):
  ```
  IEEE_LTSC_PAPI_learner_*
  ```

Rationale

The following is a rationale for these two rules for this specific transformation of the PAPI Learner datatypes.

Note: This RFC 822-like binding (PAPI Learner → DNVPs) requires 2 transformation rules. ***Other standards and different DNVP bindings may require more, fewer, or different transformation rules.***

Rationale for Rule 1

Rule 1 is the main transformation from ISO/IEC 11404 datatypes to DNVP notation. The following examples use the following definition to illustrate the transformations:

```
A: record
(
    B: integer,
    C: record
    (
        D: integer,
        E: characterstring(iso-10646-1),
    ),
    F_list: array (0..limit) of (integer),
)
```

The first three sentences, "For each data element, map all identifiers to fully-qualified field names. A fully-qualified name represents the nested structure of the data element, as described by its "aggregate datatype generator" (ISO/IEC 11404 terminology). A period (`"."`) shall separate each level of nesting in a fully-qualified field name", transform identifiers in to field names, such as:

```
A.B
```

```
A.C.D
A.C.E
```

The fourth sentence, "for array and sequence aggregates, elements are represented by repeated DNVP field names based on the identifier of the aggregate, not the index of the element", requires arrays and sequences (lists) to be represented as multiple DNVPs with the same name — permitted in RFC 822-like systems, but with a slightly different meaning (RFC 822 allows these lines to be combed, while this PAPI Learner coding binding does not permit automatic consolidation of DNVPs). For example, the data element **F** would be represented as:

```
(correct DNVP binding of F)
A.F.__begin:
A.F: xxx
A.F.__end:
A.F.__begin:
A.F: yyy
A.F.__end:
A.F.__begin:
A.F: zzz
A.F.__end:
```

but not as:

```
(incorrect DNVP binding of F)
A.F.0: xxx
A.F.1: yyy
A.F.2: zzz
```

The fifth sentence, "A field name is followed by a colon (":"), followed by the value of its associated data element", associates the field name with its value and separates the two with a colon ("**:** ").

Example:

```
A.B: 17
A.C.D: 34
A.C.E: yellow pigs
A.F: 51
A.F: 68
A.F: 85
```

Rationale for Rule 2

This rule is used for rewriting tags to use certain namespace conventions.

13.3 Consuming and interpreting dotted name-value pairs

The following rules describe the transformation of DNVPs to PAPI Learner data elements, as described by this Standard and by ISO/IEC 11404 notation.

- **Rule 1:** Transform the following XML tags (wildcard notation):
  `IEEE_LTSC_PAPI_learner_*`
  to the following XML tags (wildcard notation):
  `PAPI_learner_*`
- **Rule 2:** For each field name that is associated with an identifier defined by a PAPI Learner data element in this Standard, map each field name to the corresponding data

element identifier. The nesting of the field names represents the nesting of the data elements, i.e., the reverse of the operation in Rule #1 of subclause 13.2, Generating and Interpreting Dotted Name-Value Pairs, above. Each field value is converted to the value of the corresponding data element. An empty field value of an aggregate may represent the existence of the aggregate, but not the value of its components.

Rationale

Rationale for Rule 1

This rule transforms any namespace prefixes, as necessary. In this illustration, the namespace prefix ("`IEEE_LTSC_`") was used to reduce the possibility of namespace collisions.

Rationale for Rule 2

This rule handles the main transformation of DNVPs to data elements.

The first sentence, "for each field name that is associated with an identifier defined by a PAPI Learner data element in this Standard, map each field name to the corresponding data element identifier", (1) ignores all identifiers that are unknown to this Standard, (2) properly pairs the field names and identifiers of data elements, and (3) creates the association with data elements, but does not assign the values of the data elements..

The second sentence, "The nesting of the field names represents the nesting of the data elements, i.e., the reverse of the operation in Rule #1 of subclause 20.2, Generating and Interpreting Dotted Name-Value Pairs, above", assures that the internal structure of the DNVPs, to the extent required by this Standard, agree with the internal structure of the data elements.

The third sentence, "each field value is converted to the value of the corresponding data element", transforms the field values of the data elements, i.e., it "populates" the data elements.

The fourth sentence, "an empty field value of an aggregate may represent the existence of the aggregate, but not the value of its components", merely creates the aggregate.

In the fragment below, **`A.C`**, which has an empty value, may be used to indicate the existence of an aggregate:

```
A.B: 17
A.C:
A.C.D: 34
A.C.E: yellow pigs
```

An empty value is a useful technique when aggregates are comprised of optional data elements, i.e., the signaling of the existence of the aggregate, but the lack of aggregate components:

```
A.B: 17
A.C:
```

An empty value is not used to indicate a **`void`** type.

13.4 Representation of basic data types

The following subclauses describe the transformation of data element values to/from character representations for information interchange for use within the DNVP binding.

13.4.1 Characters and character strings

Data elements that are of type **`character`** shall be represented only as per ISO/IEC 8859-1 when encoded according to the rules of RFC 1522 .

13.4.2 Integers

Data elements that are of type **`integer`** shall be represented as per subclause 19.3.2 above.

13.4.3 Real numbers

Data elements that are of type **`real`** shall be represented as per subclause 19.3.3 above.

13.4.4 Date and time values

Data elements that are of type **`time`** shall be represented as per subclause 19.3.4 above.

13.4.5 Void types

A void type shall have no representation and shall have no encoding.

Example: The following record

```
A: record
(
    B: integer,
    C: void,
    D: characterstring(iso-10646-1),
)
```

is represented in an RFC 822-like binding as:

```
(correct DNVP binding of C)
A.B: 17
A.D: hello
```

but not as:

```
(incorrect DNVP binding of C)
A.B: 17
A.C:
A.D: hello
```

13.5 Encoding of character representations

Conforming PAPI Leaner DNVP coding bindings shall encode character representations to character values specified by ISO/IEC 8859-1. Characters outside this repertoire shall be encoded according to the rules of RFC 1522.

13.6 Handling exceptions and extensions

13.6.1 Implementation-defined behavior

The following are implementation-defined behaviors in addition to those described elsewhere in this Standard.

The following are implementation-defined behaviors in the production and consumption of DNVP codings:

- The encoding, in characters, of the newline character. Note: Implementations can avoid implementation-defined behavior by using the carriage return line feed combination characters for the newline character. See subclause 20.1.1, Basic Lexical Elements, above.
- The maximum size, in characters, of a strictly conforming PAPI Learner data instance, coded as a DNVP, that may be processed successfully.
- The time zone information for data elements of **`time`** type that have unspecified timezones.

13.6.2 Unspecified behavior

The following are unspecified behaviors in addition to those described elsewhere in this Standard.

The following is unspecified behaviors in the generation or interpretation of DNVPs:

- The order of processing data elements.

13.6.3 Undefined behavior

The following are undefined behaviors in addition to those described elsewhere in this Standard.

The following are undefined behaviors in the production or consumption of DNVPs:

- The use of field names that correspond to extended data elements.
- The use of field names that correspond to reserved data elements.
- The use of field names not specified in this DNVP coding binding.
- The use of characters outside the repertoire described in this Standard.

14 Annex E: Document development (informative)

This Annex is informative and not normative.

This section concerns the development of this document. The past (revision history, resolved issues), present (release notes, comment returns), and future (open issues) releases of this document are identified here.

14.1 Revision history

- **Draft 1, 1997-03-27**, the first draft. This draft was just the schema of the personal information (at the time, known as "Personal Information Management Systems (PIMS)").
- **Draft 2, 1997-10-07**, the second draft. This draft was largely complete and submitted as a base document for IEEE 1484.2.
- **Draft 3, 1998-04-02**, the third draft. Revised wording to incorporate comments from standards activity. Harmonized work with IMS specifications. Incorporated schema changes from prototyped implementations.
- **Draft 4, 1998-06-08**, the fourth draft. Added TCP and Java server bindings.
- **Draft 5, 1998-11-26**, the fifth draft. Added functionality and conceptual model.
- **Draft 6, 2000-06-23**, the sixth draft. Added semantics and bindings sections. Converted to IEEE format.
- **Draft 7, 2000-11-28**, the current draft. Added examples and bindings. Split document into several pieces.

14.2 Release notes for this document

The following notes apply to this release of this Standard:

- The glossary needs harmonization with the IEEE 1484.3 work.
- The codings and API work will be harmonized with IEEE 1484.14 and other activities.
- The protocols will be harmonized with IEEE 1484.15 and other activities.
- PARs need to be written for other parts of the document.

14.3 Resolved issues

The following issues have been resolved:

- Multiple syntax bindings. After much lively discussion in IEEE 1484.2, there is agreement that multiple syntax bindings are necessary.
- Functionality and conceptual model. This work has benefited from much discussion in IMS.
- Conceptual model and conformance wording has been defined.
- Informative material has been moved to the end of the document.
- Significantly reduced size of document.

14.4 Open issues

The following issues are outstanding:

- Interoperability negotiation. When two systems interoperate, they need to determine and resolve incompatibilities in their data models.
- Completing remaining coding, API, and protocol definitions.
- Agreement on semantic codings. Need to have naming conventions for identifying semantic coding conventions, e.g., creating the name "US-NY-LETTER-GRADE". Need to consider mechanisms for translating from coding to another.
- Other syntax bindings. What other syntax bindings are necessary? What tools are available to convert syntax bindings and how are they invoked?
- Rationale. Need more words on rationale for particular choices, such as multiple syntax bindings. This will be incorporated into a separate annex.

14.5 Comments on this document

All comments are appreciated. Please return all comments on this release of this document by **Sunday, 2000-12-31 23:00 UTC**. Deliver all comments to the IEEE 1484.2 Learner Model Working Group by sending E-mail to:

`ltsc-learner@majordomo.ieee.org`

To subscribe to the working group mailing list, send the one-line message

`subscribe ltsc-learner`

to the E-mail address "`majordomo@majordomo.ieee.org`".

The Technical Editor may be contacted at any one of the following:

```
Telephone: +1 212 486 4700
Fax: +1 212 759 1605
E-mail: frank@farance.com

Frank Farance
Farance/Edutool
Island Box 256
New York, NY
10044-0205
USA
```