P1849™/D01
Draft Standard for XES - eXtensible Event Stream - for achieving interoperability in event logs and event streams

Sponsor

New Standards Committee
of the
IEEE IEEE Computational Intelligence Society

Approved <Date Approved>

IEEE-SA Standards Board

IEEE Standards Department
445 Hoes Lane
Piscataway, NJ 08854, USA

Abstract: The XES standard defines a grammar for a tag-based language whose aim is to provide designers of information systems with a unified and extensible methodology for capturing systems' behaviors by means of event logs and event streams. This standard includes a "XML Schema" describing the structure of an XES event log/stream and a "XML Schema" describing the structure of an extension of such a log/stream. Moreover, the standard includes a basic collection of so-called "XES extension" prototypes that provide semantics to certain attributes as recorded in the event log/stream.

Keywords: event log, event stream, system behavior, extensions, XML.

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

PDF: ISBN 978-0-XXXX-XXXX-X STDXXXXX
Print: ISBN 978-0-XXXX-XXXX-X STDPDXXXXX

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading "Important Notice" or "Important Notices and Disclaimers Concerning IEEE Standards Documents."

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association ("IEEE-SA") Standards Board. IEEE ("the Institute") develops its standards through a consensus development process, approved by the American National Standards Institute ("ANSI"), which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. IEEE standards documents are supplied "AS IS" and "WITH ALL FAULTS."

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official statements

Comments on standards

Laws and regulations

Copyrights

Photocopies

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every ten years. When a document is more than ten years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE-SA Website at http://ieeexplore.ieee.org/xpl/standards.jsp or contact IEEE at the address listed previously. For more information about the IEEE-SA or IEEE's standards development process, visit the IEEE-SA Website at http://standards.ieee.org.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE-SA Website at the following URL: http://standards.ieee.org/findstds/errata/index.html. Users are encouraged to check this URL for errata periodically.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE-SA Website at http://standards.ieee.org/about/sasb/patcom/patents.html. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants

At the time this draft standard was completed, the XES Working Group had the following membership:

Wil van der Aalst, *Chair*
Christian Günther, *Vice Chair*

JC Bose
Josep Carmona
Marlon Dumas
Frank van Geffen
Sukriti Goel
Antonella Guzzo
Rania Khalaf
Rudolf Kuhn
Teemu Lehto
Felix Mannhardt
Marco Montali
Michael zur Muehlen
Zbigniew Paszkiewicz
Hajo Reijers
Alexander Rinke
Michal Rosik
Anne Rozinat
Pnina Soffer
Minseok Song
Keith Swenson
Walter Vanherle
Eric Verbeek
Lijie Wen
Moe Wynn

The following members of the <individual/entity> balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

[To be supplied by IEEE]

Balloter1
Balloter2
Balloter3
Balloter4
Balloter5
Balloter6
Balloter7
Balloter8
Balloter9

When the IEEE-SA Standards Board approved this standard on <Date Approved>, it had the following membership:

[To be supplied by IEEE]

<Name>, *Chair*
<Name>, *Vice Chair*
<Name>, *Past Chair*
Konstantinos Karachalios, *Secretary*

SBMember1
SBMember2
SBMember3
SBMember4
SBMember5
SBMember6
SBMember7
SBMember8
SBMember9

*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

<Name>, *DOE Representative*
<Name>, *NIST Representative*

<Name>
IEEE-SA Content Production and Management
<Name>
IEEE-SA Technical Program Operations

Introduction

This introduction is not part of P1849/D01, Draft Standard for XES - eXtensible Event Stream - for achieving interoperability in event logs and event streams .

Event logs contain information on how processes have evolved in running systems. As more and more systems capture such information, there is a need to be able to transfer this information from these running systems to a site where the information can be analyzed, either automatically by software from the computational intelligence field, or manually (at least in part) using such software.

This Standard addresses this need by defining an eXtensible Event Stream (XES) structure for such event logs.

Furthermore, this Standard defines the World Wide Web Consortium (W3C) Extensible Markup Language (XML) structure and constraints on the contents of XML 1.1 documents that can be used to represent XES instances, and a likewise structure (called XESEXT) that can be used to represent so-called extensions to this structure.

The purpose of this Standard is to allow the generation and analysis of event logs using XML. This Standard uses the W3C XML Schema definition language as the encoding, which allows for interoperability and the exchange of XES XML instances between various systems.

Contents

Draft Standard for XES - eXtensible Event Stream - for achieving interoperability in event logs and event streams

1. Overview

The scope and purpose of this Standard are discussed in 1.1 and 1.2.

1.1 Scope

This Standard defines World Wide Web Consortium (W3C) Extensible Markup Language (XML) structure and constraints on the contents of XML 1.1 documents that can be used to represent extensible event stream (XES) instances. A XES instance corresponds to a file-based event log or a formatted event stream that can be used to transfer event-driven data in a transparent manner from a first site to a second site. Typically, the first site will be the site generating this event-driven data (for example, workflow systems, case handling systems, procurement systems, devices like wafer steppers and X-ray machines, and hospitals) while the second site will be the site analyzing this data (for example, by data scientists and/or advanced software systems).

1.2 Purpose

The purpose of this Standard is to provide a generally-acknowledged format for the interchange of event data between tools and applications domains. As such, this Standard aims to fix the syntax and the semantics of the event data which, for example, is being transferred from the site generating this data to the site analyzing this data. As a result of this Standard, if the event data is transferred using the syntax as described by this Standard, its semantics will be well-understood and clear at both sites.

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

- ISO 639-1, Code for the Representation of Names of Languages—Part 1: Alpha-2 code.
- ISO 639-2, Codes for the Representation of Names of Languages—Part 2: Alpha-3 code.
- ISO 4217:2008, Codes for the Representation of Currencies and Funds.
- ISO/IEC 9834-8:2014, Information technology—Procedures for the operation of object identifier registration authorities—Part 8: Generation of universally unique identifiers (UUIDs) and their use in object identifiers.
- W3C Recommendation (28 October 2004), XML Schema Part 1: Structures, Second Edition.
- W3C Recommendation (28 October 2004), XML Schema Part 2: Datatypes, Second Edition.
- W3C Recommendation (4 February 2004), Namespaces in XML 1.1.

3. Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause. [1]

component: A XES element that may contain XES attributes, that is, a log, a trace, an event, or an attribute.

4. XES metadata structure

4.1 Hierarchical components

4.1.1 Log component

A *log component* represents information that is related to a specific process. Examples for processes are handling insurance claims, using a complex X-ray machine, and browsing a website. A log shall contain a (possibly empty) collection of traces followed by a (possibly empty) list of events. The order of the events in this list is important, as it signifies the order in which the events have been observed.

[1]*IEEE Standards Dictionary Online* subscription is available at:
http://www.ieee.org/portal/innovate/products/standard/standards_dictionary.html.

If the log contains only events and no traces, then the log is also called a *stream*.

4.1.2 Trace component

A *trace component* represents the execution of a single case, that is, of a single execution (or enactment) of the specific process. A trace shall contain a (possible empty) list of all events that are related to a single case. The order of the events in this list is important, as it signifies the order in which the events have been observed.

4.1.3 Event component

An *event component* represents an atomic granule of activity that has been observed. If the event occurs in some trace, then it is clear to which case the event belongs. If the event does not occur in some trace, that is, if it occurs in the log, then we need ways to relate events to cases. For this, we will use the combination of a trace classifier and an event classifier, see 4.4.

4.2 Attribute component

Information on any component (log, trace, or event) is stored in *attribute components*. Attributes describe the enclosing component, which may contain an arbitrary number of attributes. However, no two attributes of the same component may share the same key, that is, every key may occur only once in a single component.

For providing maximum flexibility, this Standard allows for *nested* attributes, that is, attributes that themselves have child attributes. While this feature is necessary for efficient encoding of certain information types, it is *optional for tools to implement nested attributes*, that is, the feature is not strictly required in order to be compliant to this Standard. Nevertheless, a tool that does not support nested attributes shall be able to read documents which feature nested attributes. These tools shall transparently ignore and discard any nested attributes, and, where feasible, alert the user to the fact that some information may not be available.

An attribute may be either *elementary* or *composite*.

4.2.1 Elementary attributes

An *elementary attribute* is an attribute that contains an elementary (single, basic) value. In this Standard, an elementary attribute can be a *string attribute*, a *date and time attribute*, an *integer number attribute*, a *real number attribute*, a *Boolean attribute*, or an *ID attribute*.

4.2.1.1 String attributes

Valid values for a string attribute are values that conform to the `xs:string` datatype.

4.2.1.2 Date and time attributes

Valid values for a date and time attribute are values that conform to the `xs:dateTime` datatype.

4.2.1.3 Integer number attributes

Valid values for an integer number attribute are values that conform to the `xs:long` datatype.

4.2.1.4 Real number attributes

Valid values for a real number attribute are values that conform to the `xs:double` datatype.

4.2.1.5 Boolean attributes

Valid values for a Boolean attribute are values that conform to the `xs:boolean` datatype.

4.2.1.6 ID attributes

Valid values for an ID attribute are values that conform to the ID datatype, that is, all string representations of UUIDs.

4.2.2 Composite attributes

A *composite attribute* is an attribute that may contain multiple values. In this Standard, a composite attribute is a *list attribute*.

4.2.2.1 List attribute

Valid values for the list datatype are all lists (series) of attribute values. The order between the child attributes in this list is important. In contrast to attributes enclosed in a component, attributes enclosed in a list may share the same key.

4.3 Global attributes

A log shall hold a (possibly empty) list of global attribute declarations. A global attribute declaration shall have a valid *key*, a valid *datatype*, and a valid *value* for the datatype. A global attribute declaration can either be for events or for traces.

Global attributes are a required feature for compliance to this Standard. Nevertheless, a defensive approach is recommended with respect to global attributes, as there is no way to undo a global declaration.

4.3.1 Global event attributes

Global event attributes are event attributes that are understood to be *available* and *properly defined* for each event in the log (be it in a trace or not). As a result, every event in the log shall contain an attribute with the given key and the given datatype, but possibly with a different valid value. The value provided for a global event attribute declaration is *only significant* in case an event *needs to be created* (for some reason) for which *no value is provided* for that attribute. In that case, the value of the declaration shall be used as the value for the attribute. In all other cases, the value of the declaration is insignificant, and shall not be used.

4.3.2 Global trace attributes

Global trace attributes are trace attributes that are understood to be *available* and *properly defined* for each trace in the log. As a result, every trace shall contain an attribute with the given key and the given datatype, but possibly with a different valid value. The value provided for a global attribute declaration is *only significant* in case a trace *needs to be created* (for some reason) for which *no value is provided* for that attribute. In that case, the value of the declaration shall be used as the value for the attribute. In all other cases, the value of the declaration is insignificant, and shall not be used.

4.4 Classifiers

In this Standard, there are per se no predefined attributes with any well-understood meaning. Instead, a log shall hold a (possibly empty) list of classifiers. *These classifiers are a mandatory feature of this Standard*.

A classifier assigns to each event an *identity*, which makes it comparable to others (via their assigned identity). Examples of such identities include the descriptive name of the event, the descriptive name of the case the event relates to, the descriptive name of the cause of the event, and the descriptive name of the case related to the event.

A classifier can either be an *event classifier* or a *trace classifier*.

In case the log contains events that do not occur in a trace, then it is necessary to be able to relate these events to cases. For this reason, we assume that one of the existing event classifiers provides the descriptive name of its case. Two events for which this classifier result in the same identity, belong to the same case. Furthermore, we assume that one of the trace classifiers provides the descriptive name for the case. If this classifier and the event classifier mentioned earlier return the same identity, then the corresponding event belongs to the same case as the corresponding trace. As such, the event can be appended to this trace. If no matching trace exists, a new trace may be started with the event.

4.4.1 Event classifiers

An event classifier shall be defined via an ordered list of attribute keys. The identity of the event shall be derived from the actual values of the attributes with these keys. An attribute whose key appears in an event classifier list shall be declared as a global event attribute before the event classifier is defined, as the actual value for the attribute is required by the event classifier.

4.4.2 Trace classifiers

A trace classifier shall be defined via an ordered list of attribute keys. The identity of the trace shall be derived from the actual values of the attributes with these keys. An attribute whose key appears in a trace classifier list shall be declared as a global trace attribute before the trace classifier is defined, as the actual value for the attribute is required by the trace classifier.

4.4.3 Event ordering

Within the context of a single trace, the ordering of events shall be important: An event that occurs in a log (be it in a trace or in the log itself) before another event that is related to the same trace, shall be assumed to have occurred before that other event. However, the notion of a trace also depends on the trace and event classifiers selected by the user. As such, the notion of a trace is not necessarily predefined for a log. As a result, the notion of the event ordering may be affected by the choice of classifiers.

Whatever classifier is selected by the user, the ordering in the log shall be maintained: The first event that belongs to some trace shall be the first event encountered in the log (be it in a trace or in the log itself), etc. As an example, consider the event log that contains (1) a trace containing events e_{11} and e_{12}, (2) a trace containing an event e_{21}, (3) a trace containing events e_{31}, e_{32}, and e_{33}, and (4) an event e_4. Now, assume that the user has selected an event classifier and a trace classifier that causes the events e_{11}, e_{31}, e_{33}, and e_4 to belong to the same trace. Then the ordering in this (classified) trace shall be e_{11}, e_{31}, e_{33}, and e_4.

4.5 Extensions

This Standard does not define a specific set of attributes per component. As such, the semantics of the data attributes these elements do contain must necessarily be ambiguous, hampering the interpretation of that data.

This ambiguity is resolved by the concept of *extensions* in this Standard. An extension defines for every type of component a (possibly empty) set of attributes. The extension provides points of reference for interpreting these attributes, and, thus, their components. Extensions therefore are *primarily a vehicle for attaching semantics to a set of defined attributes per component*.

Extensions have many possible uses. One important use is to introduce a set of commonly understood attributes which are vital for a specific perspective or dimension of event log analysis (and which may even not have been foreseen at the time of developing this Standard). See Clause 7 for the current set of standard extensions.

Other uses include the definition of generally-understood attributes for a specific application domain (for example, medical attributes for hospital processes), or for supporting special features or requirements of a specific application.

An extension shall have a descriptive *name*, a *prefix*, and a *URI*. The prefix is the prefix of all attributes defined by the extension. This means, the keys of all attributes defined by the extension will be prepended with this prefix and colon separation character (like a namespace in XML). The URI is a unique URI which points to the definition of the extension.

The definition of the extension shall contain for every component a (possibly empty) list of attribute declarations. An attribute declaration shall contain the *key* of the attribute, the *datatype* of the attribute, and a (possible empty) list of *aliases*. An alias shall contain the *descriptive text* for the attribute (that is, the commonly understood semantics for the attribute) and the *language code* of the language of this descriptive text.

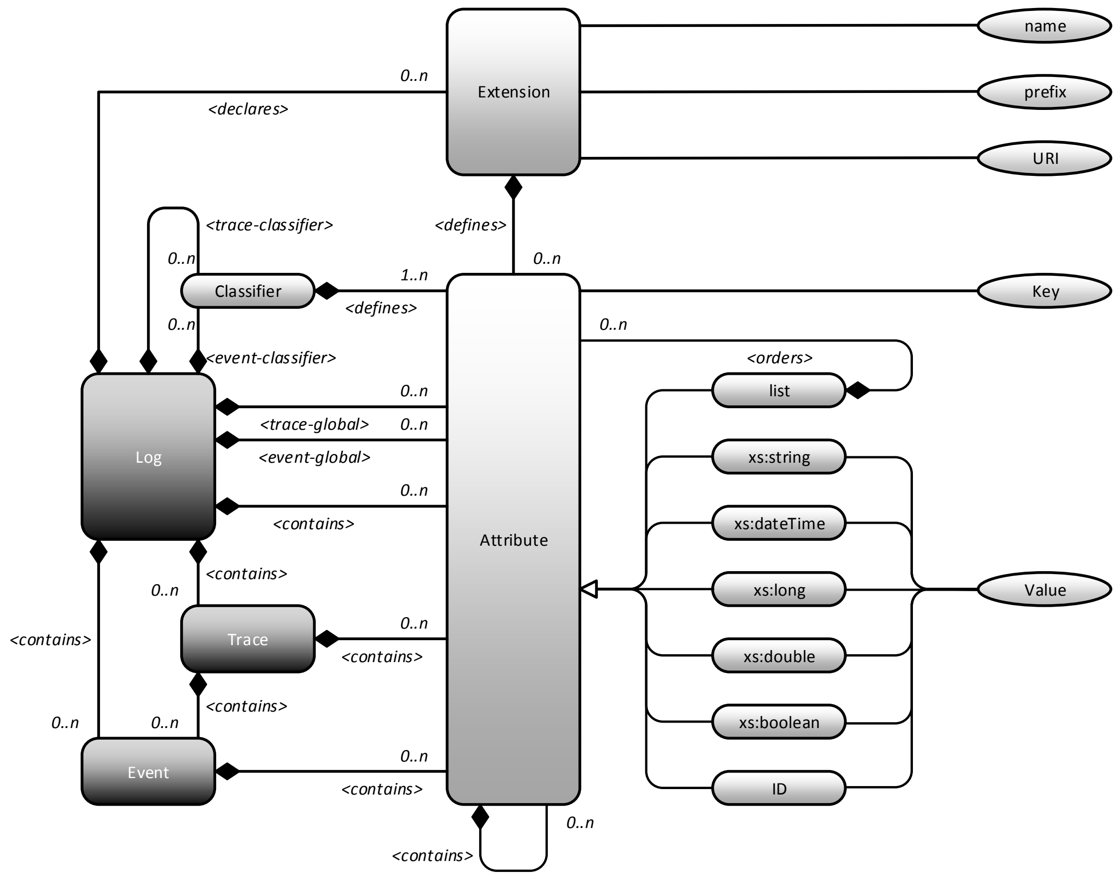

Figure 1 Overview of the XES metadata structure

5. XES XML serialization

5.1 Log element

Captures the log component from the XES metadata structure.

— XML name: `log`.

5.1.1 Elements

The following (sub) elements should appear in the specified order.

XML name	*XES element*	*Min*	*Max*	*Description*
`extension`	Extension, see 5.5.	0	∞	An extension declaration for the log.
`global`	Global, see 5.6.	0	∞	A global (event or trace) attribute for the log.

`classifier`	Classifier, see 5.7.	0	∞	A classifier (even tor trace) definition for the log.
attribute	Attribute, see 5.4.	0	∞	An attribute for the log. May be elementary or composite.
`trace`	Trace, see 5.2.	0	∞	A trace for the log.
`event`	Event, see 5.3.	0	∞	An event for the log.

5.1.2 Attributes

Attribute key	*Attribute type*	*Status*	*Description*
`xes.version`	`xs:decimal`	Required	The version of the XES standard the document conforms to (e.g., `2.0`).
`xes.features`	`xs:token`	Required	A whitespace-separated list of optional XES features this document makes use of (e.g., `nested-attributes`). If not optional features are used, this attribute should have an empty value.

5.2 Trace element

Captures the trace component from the XES metadata structure.

— XML name: `trace`.

5.2.1 Elements

The following (sub) elements should appear in the specified order.

XML name	*XES element*	*Min*	*Max*	*Description*
attribute	Attribute, see 5.4.	0	∞	An attribute for the trace. May be elementary or composite.
`event`	Event, see 5.3.	0	∞	An event for the trace.

5.2.2 Attributes

N/A.

5.3 Event element

Captures the event component from the XES metadata structure.

— XML name: `event`.

5.3.1 Elements

The following (sub) elements should appear in the specified order.

XML name	*XES element*	*Min*	*Max*	*Description*
attribute	Attribute, see 5.4.	0	∞	An attribute for the event. May be elementary or composite.

5.3.2 Attributes

N/A.

5.4 Attribute element

Captures the attribute component from the XES metadata structure. Can be any of the following:

XML name	*Datatype*	*Value*	*Description*
`string`	`xs:string`	Elementary	A string value.
`date`	`xs:dateTime`	Elementary	A data and time value.
`int`	`xs:long`	Elementary	An integer number value.
`float`	`xs:double`	Elementary	A real number value.
`boolean`	`xs:boolean`	Elementary	A Boolean value.
`id`	ID	Elementary	A UUID value.
`list`	Attribute list	Composite	A sorted list of attributes.

5.4.1 Elements for elementary attributes

The following (sub) elements should appear in the specified order.

XML name	*XES element*	*Min*	*Max*	*Description*
attribute	Attribute, see 5.4.	0	∞	An attribute for the attribute, that is, a meta-attribute. May be elementary or composite.

5.4.2 Elements for composite attributes

The following (sub) elements should appear in the specified order.

XML name	*XES element*	*Min*	*Max*	*Description*
attribute	Attribute, see 5.4.	0	∞	An attribute for the attribute, that is, a meta-attribute. May be elementary or composite.
`values`	Attribute list	1	1	The ordered list of attributes that constitute the composite value of the attribute. Attributes in this list may be elementary and composite, and attributes in this list may share the same key.

5.4.3 Attributes for elementary attributes

Attribute key	*Attribute type*	*Status*	*Description*
`key`	`xs:string`	Required	The key of the attribute.
`value`	`xs:string`	Required	The value (as a string) of the elementary attribute.

5.4.4 Attributes for composite attributes

Attribute key	*Attribute type*	*Status*	*Description*
`key`	`xs:string`	Required	The key of the attribute.

5.5 Extension element

Captures the extension declaration from the XES metadata structure.

— XML name: `extension`.

5.5.1 Elements

N/A.

5.5.2 Attributes

Attribute key	*Attribute type*	*Status*	*Description*
`name`	`xs:NCName`	Required	The name of the extension.
`prefix`	`xs:NCName`	Required	The prefix of the extension.
`uri`	`xs:anyURI`	Required	The URI from where the definition of this extension (should be a file that conforms to the

		XESEXT format) can be retrieved.

5.6 Global element

Captures the global attribute declaration from the XES metadata structure.

— XML name: `global`.

5.6.1 Elements

XML name	*XES element*	*Min*	*Max*	*Description*
attribute	Attribute, see 5.4.	1	1	A global attribute declaration for the log. Depending on whether the attribute is declared global for events or traces, every event or trace in the log should have this attribute as sub element.

The attribute value provided with this global declaration should only be used if a new event or trace needs to be inserted in the log for which no other valid value for this attribute can be obtained. In that case, and only in that case, may the value of this attribute be used.

5.6.2 Attributes

Attribute key	*Attribute type*	*Status*	*Description*
`scope`	`xs:NCName`	Optional	Either `event` or `trace`, to denote whether this attribute is declared global for events or traces. The default is `event`.

5.7 Classifier element

Captures the classifier definition from the XES metadata structure.

— XML name: `classifier`.

5.7.1 Elements

N/A.

5.7.2 Attributes

Attribute key	*Attribute type*	*Status*	*Description*

`name`	`xs:NCName`	Required	The name of the classifier.
`scope`	`xs:NCName`	Optional	Either `event` or `trace`, to denote whether this classifier can be used to classify events or traces. The default is `event`.
`keys`	`xs:token`	Required	The white-space-separated list of attribute keys that constitute this classifier.

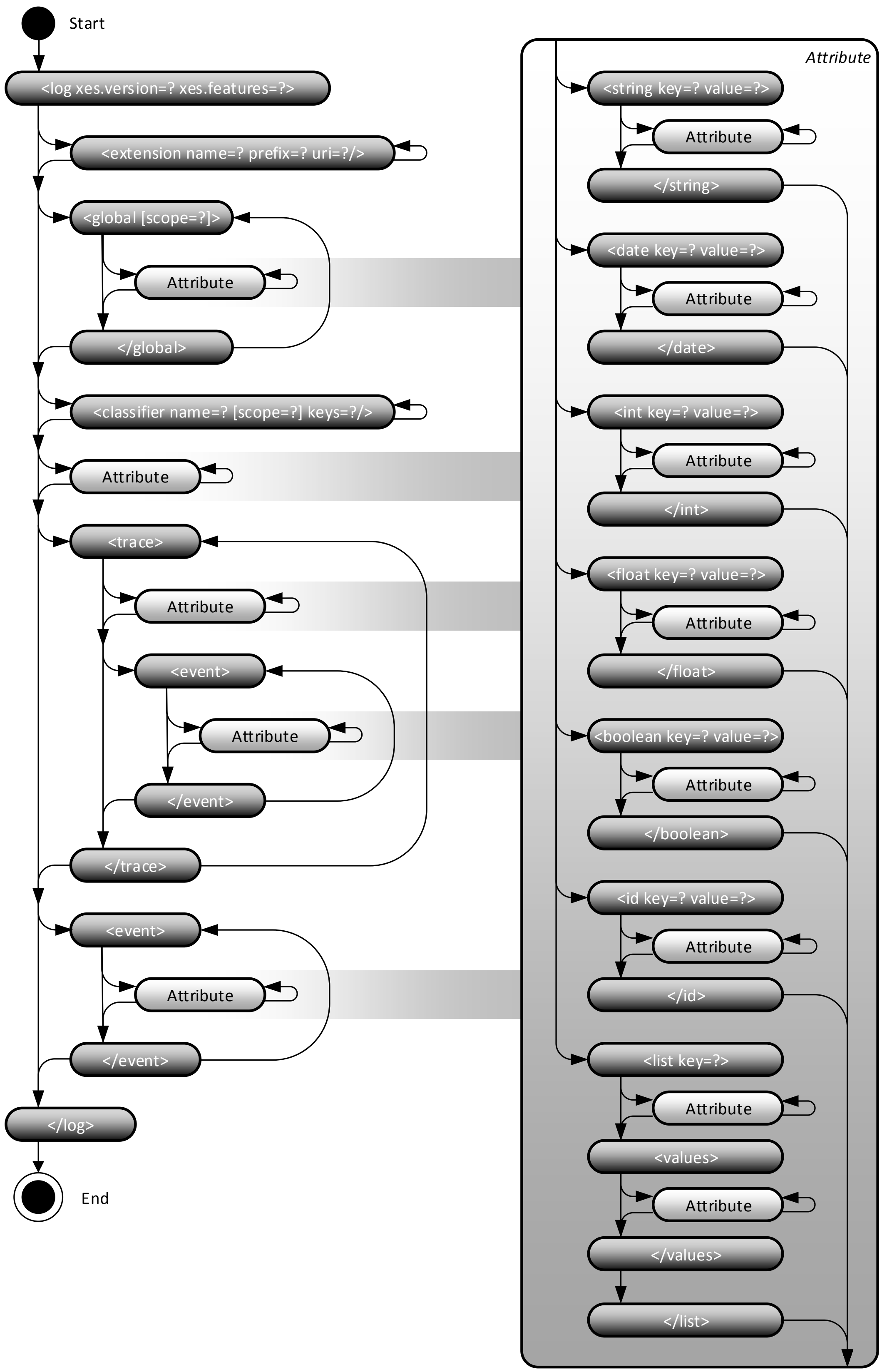

Figure 2 State machine flow diagram for XES XML serialization

6. XESEXT XML Serialization

6.1 XESExtension element

Captures the extension definition from the XES metadata structure.

— XML name: `xesextension`.

6.1.1 Elements

The following (sub) elements should appear in the specified order.

XML name	*XESEXT element*	*Min*	*Max*	*Description*
`log`	Log, see 6.2.	0	1	Attribute definitions for logs.
`trace`	Trace, see 6.3.	0	1	Attribute definitions for traces.
`event`	Event, see 6.4	0	1	Attribute definitions for events.
`meta`	Meta, see 6.5	0	1	Attribute definitions for attributes.

6.1.2 Attributes

Attribute key	*Attribute type*	*Status*	*Description*
`name`	`xs:NCName`	Required	The name of the extension.
`prefix`	`xs:NCName`	Required	The prefix to be used for this extension.
`uri`	`xs:anyURI`	Required	The URI where this extension can be retrieved from.

6.2 Log element

Captures the log extension definition from the XES metadata structure.

— XML name: `log`.

6.2.1 Elements

The following (sub) elements should appear in the specified order.

XML name	*XESEXT element*	*Min*	*Max*	*Description*

attribute	Attribute, see 6.6.	0	∞	Attribute definition for logs.

6.2.2 Attributes

N/A.

6.3 Trace element

Captures the trace extension definition from the XES metadata structure.

— XML name: `trace`.

6.3.1 Elements

The following (sub) elements should appear in the specified order.

XML name	*XESEXT element*	*Min*	*Max*	*Description*
attribute	Attribute, see 6.6.	0	∞	Attribute definition for traces.

6.3.2 Attributes

N/A.

6.4 Event element

Captures the event definition from the XES metadata structure.

— XML name: `event`.

6.4.1 Elements

The following (sub) elements should appear in the specified order.

XML name	*XESEXT element*	*Min*	*Max*	*Description*
attribute	Attribute, see 6.6.	0	∞	Attribute definition for events.

6.4.2 Attributes

N/A.

6.5 Meta element

Captures the meta (attribute) extension definition from the XES metadata structure.

— XML name: `meta`.

6.5.1 Elements

The following (sub) elements should appear in the specified order.

XML name	*XESEXT element*	*Min*	*Max*	*Description*
attribute	Attribute, see 6.6.	0	∞	Attribute definition for attributes.

6.5.2 Attributes

N/A.

6.6 Attribute element

Captures the attribute extension element from the XES metadata structure. Can be any of the following:

XML name	*Description*
`string`	A string valued attribute.
`date`	A date and time valued attribute.
`int`	An integer number valued attribute.
`float`	A real number valued attribute.
`boolean`	A Boolean valued attribute.
`id`	A UUID valued attribute.
`list`	A list valued attribute.

6.6.1 Elements

The following (sub) elements should appear in the specified order.

XML name	*XESEXT element*	*Min*	*Max*	*Description*
`alias`	Attribute, see 6.7.	0	∞	Aliases for this attribute.

6.6.2 Attributes

Attribute key	*Attribute type*	*Status*	*Description*
`key`	`xs:string`	Required	The key of the attribute.

6.7 Alias

Captures the alias extension element from the XES metadata structure.

6.7.1 Elements

N/A.

6.7.2 Attributes

Attribute key	*Attribute type*	*Status*	*Description*
`mapping`	`xs:NCName`	Required	The language code (using the ISO 639-1 and 639-2 Standards) for this alias.
`name`	`xs:string`	Required	The semantics of this attribute described using the language with the given code.

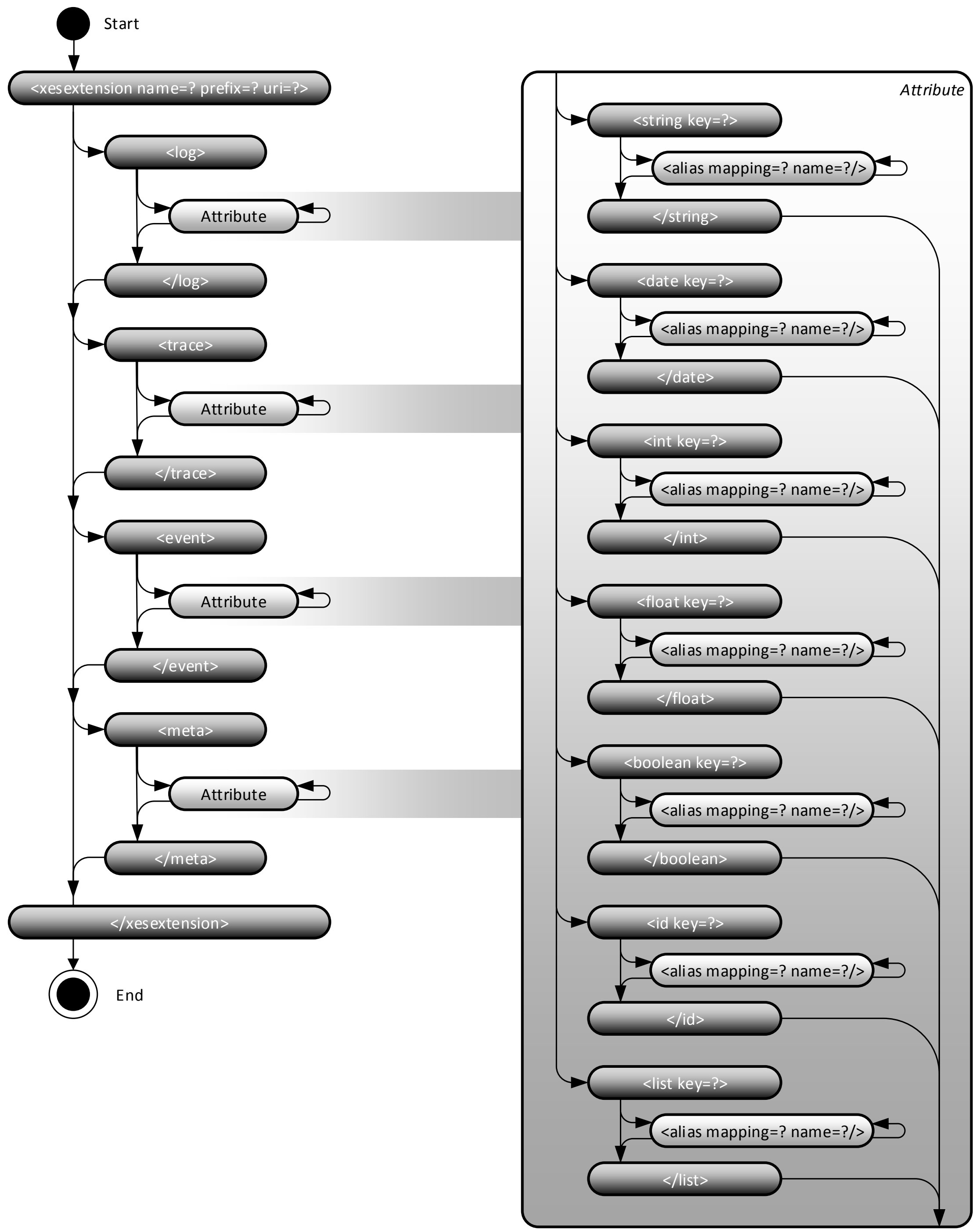

Figure 3 State machine diagram for XESEXT XML serialization

7. XES standard extensions

XES shall recognize and treat all extensions as equal, independent from their source. This allows users of the format to extend it, in order to fit any purpose or domain setting. However, there are recurring requirements for information stored in event logs, which demand a fixed and universally understood semantics. For this purpose, a number of extensions have been standardized. When creating logs for a specific domain, or also when designing log-analyzing techniques, one should consider using these standardized extensions, since they allow for a wider level of understanding of the contents of event logs.

In the following, the currently standardized extensions to the XES formats are introduced.

7.1 Concept extension

The Concept extension defines, for all levels of the XES type hierarchy, an attribute which stores the generally understood name of type hierarchy elements.

Name	*Prefix*	*URI*
Concept	`concept`	`http://www.xes-standard.org/concept.xesext`

7.1.1 Attributes

Name	*Key*	*Components*	*Datatype*	*Description*
Name	`name`	Log Trace Event	`xs:string`	Stores a generally understood name for any component type. For streams and logs, the name attribute may store the name of the process having been executed. For traces, the name attribute usually stores the case ID. For events, the name attribute represents the name of the event, e.g. the name of the executed activity represented by the event.
Instance	`instance`	Event	`xs:string`	This represents an identifier of the activity instance whose execution has generated the event. This way, multiple instances (occurrences) of the same activity can be told apart.

7.2 Lifecycle extension

The Lifecycle extension specifies for events the lifecycle transition they represent in a transactional model of their generating activity. This transactional model can be arbitrary. However, the Lifecycle extension also specifies a standard transactional model for activities, see the figure that follows. The use of this extension is appropriate in any setting where events denote lifecycle transitions of higher-level activities.

Name	*Prefix*	*URI*
Lifecycle	`lifecycle`	`http://www.xes-standard.org/lifecycle.xesext`

7.2.1 Attributes

Name	*Key*	*Components*	*Datatype*	*Description*
Model	`model`	Log	`xs:string`	This attribute refers to the lifecycle transactional model used for all events in the log. If this attribute has a value of `standard`, the standard lifecycle transactional model of this extension is assumed. If it is has a value of `bpaf`, the BPAF lifecycle transactional model is assumed.
Transition	`transition`	Event	`xs:string`	The transition attribute is defined for events, and specifies the lifecycle transition of each event.
State	`state`	Event	`xs:string`	The state attribute is defined for events, and specifies the lifecycle state of each event.

7.2.2 BPAF lifecycle transactional model

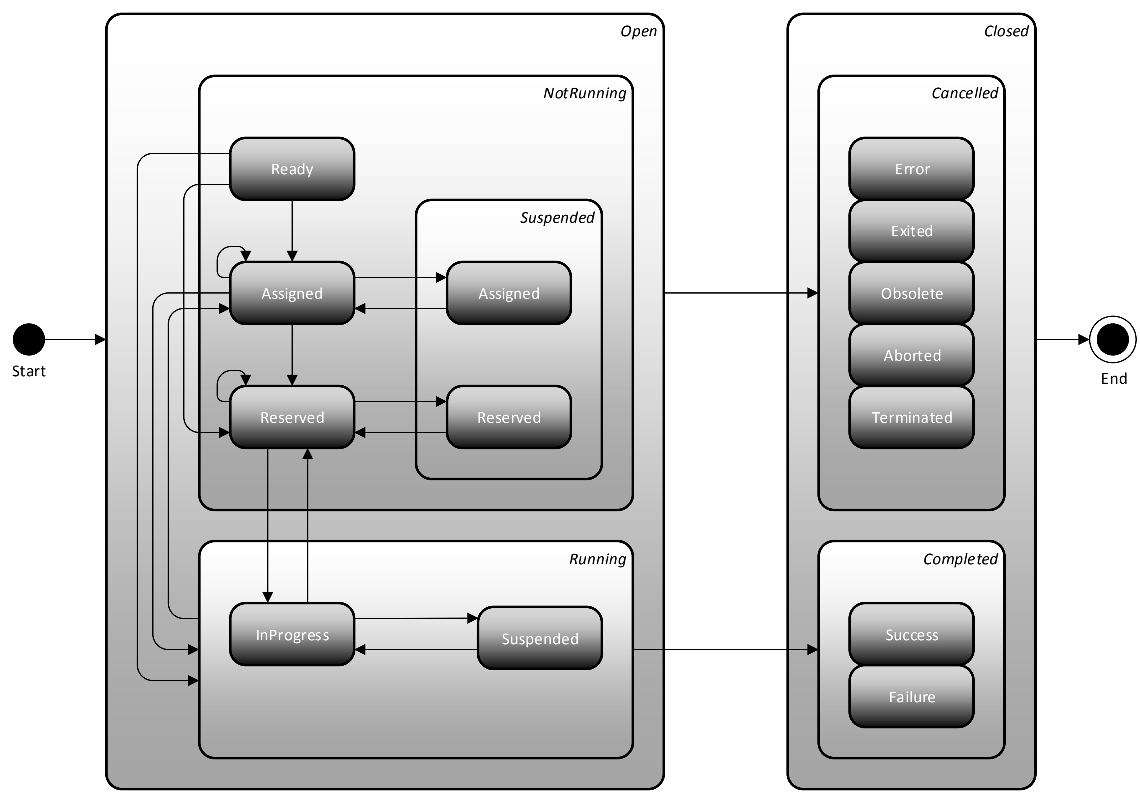

Figure 4 State machine for the BPAF transactional model

Note that the transition shown in this figure are the most typical transitions, but manual interventions and different system implementations may lead to additional transitions not depicted in the model.

State	*Description*
Closed	The activity is closed for execution.
Closed.Cancelled	The execution of the activity is cancelled.
Closed.Cancelled.Aborted	The execution of the activity is aborted.
Closed.Cancelled.Error	The execution of the activity is in error.
Closed.Cancelled.Exited	The execution of the activity is exited manually.
Closed.Cancelled.Obsolete	The execution of the activity is obsolete (e.g. in case of a timeout).
Closed.Cancelled.Terminated	The execution of the activity is forcibly terminated.
Completed	The execution of the activity is completed, i.e., it has ended naturally.
Completed.Failed	The execution of the activity is completed, but the result is unsuccessful (from a business perspective).
Completed.Success	The execution of the activity is completed, and the result is successful (from a business perspective).
Open	The activity is open for execution.
Open.NotRunning	The execution of the activity is not on-going.
Open.NotRunning.Assigned	The activity is on the worklists of resources.
Open.NotRunning.Reserved	The activity has been selected by a resource to work on.
Open.NotRunning.Suspended.Assigned	The activity is on the worklists of resources, but is barred from execution.
Open.NotRunning.Suspended.Reserved	The activity has been selected by a resource, but is barred from execution.
Open.Running	The execution of the activity is on-going.
Open.Running.InProgress	The execution of the activity is in progress.
Open.Running.Suspended	The execution of the activity is on-going, but not in progress.

7.2.3 Standard lifecycle transition model

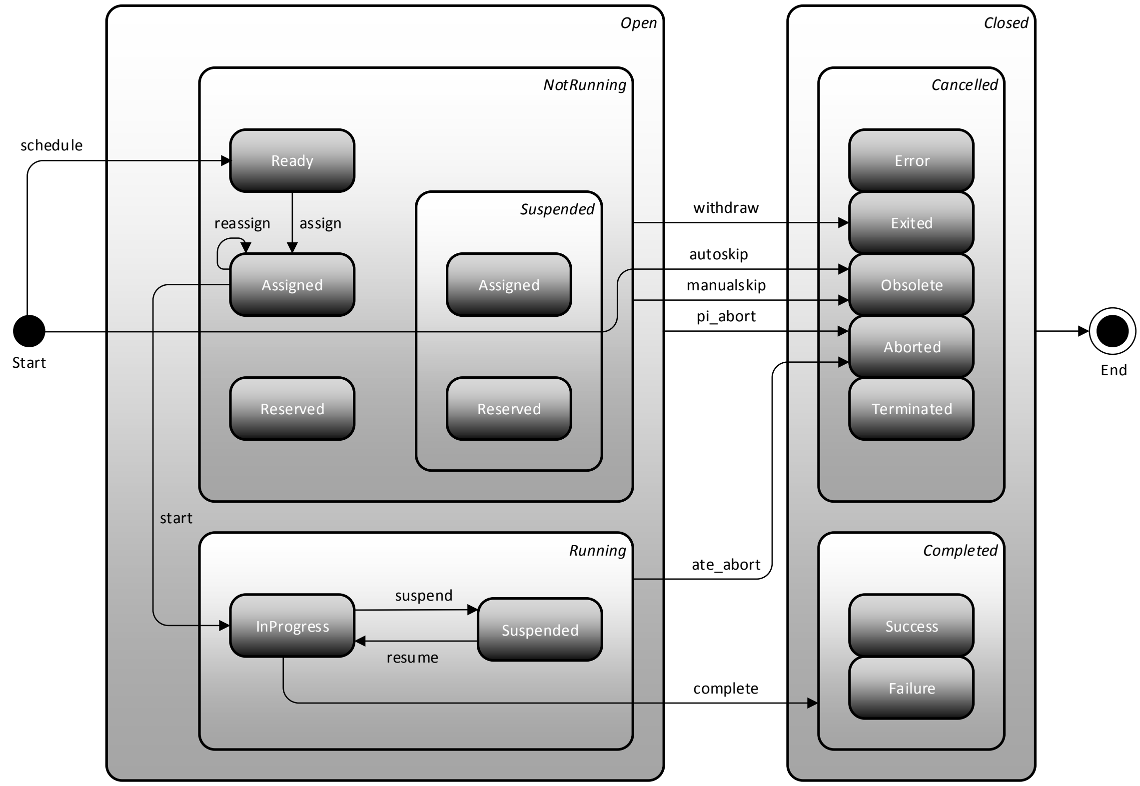

Figure 5 State machine for the standard transactional model

In contrast with the BPAF transactional model, the standard lifecycle model uses the transitions instead of the states to denote the new state of an activity.

Transition	*From State*	*To State*	*Description*
assign	Open. NotRunning. Ready	Open. NotRunning. Assigned	The activity is assigned for execution to a resource.
ate_abort	Open. Running	Closed. Cancelled. Aborted	The execution of the activity is aborted, the execution of the corresponding case is not aborted. As a result, execution of the activity has failed.
autoskip	Start	Closed. Cancelled. Obsolete	The execution of the activity is skipped by the system. As a result, execution of the activity has succeeded.
complete	Open. Running. InProgress	Closed. Completed	The execution of the activity is completed. As a result, execution of the activity has succeeded.
manualskip	Open.	Closed. Cancelled.	The execution of the activity is skipped by the user. As a

	NotRunning	Obsolete	result, execution of the activity has succeeded.
pi_abort	Open	Closed. Cancelled. Aborted	The execution of the activity is aborted, the execution of the corresponding case is also aborted. As a result, execution of the activity has failed.
reassign	Open. NotRunning. Assigned	Open. NotRunning. Assigned	The activity has been reassigned for execution to another resource
resume	Open. Running. Suspended	Open. Running. InProgress	The execution of the activity is resumed.
schedule	Start	Open. NotRunning. Ready	The activity is scheduled for execution.
start	Open. NotRunning. Assigned	Open. Running. InProgress	The execution of the activity is started.
suspend	Open. Running. InProgress	Open. Running. Suspended	The execution of the activity is suspended.
unknown	Any	Any	Any lifecycle transition not captured by any of the other transitions.
withdraw	Open. NotRunning	Closed. Cancelled. Exited	The assignment of the activity is revoked. As a result, execution of the activity has failed.

7.3 Organizational extension

The organizational extension is useful for domains, where events can be caused by human actors, who are somewhat part of an organizational structure. This extension specifies three attributes for events, which identify the actor having caused the event, and his position in the organizational structure.

Name	*Prefix*	*URI*
Organizational	`org`	`http://www.xes-standard.org/org.xesext`

7.3.1 Attributes

Name	*Key*	*Components*	*Datatype*	*Description*
Resource	`resource`	Event	`xs:string`	The name, or identifier, of the resource that triggered the event.

Role	`role`	Event	`xs:string`	The role of the resource that triggered the event, within the organizational structure.
Group	`Group`	Event	`xs:string`	The group within the organizational structure, of which the resource that triggered the event is a member.

7.4 Time extension

In almost all applications, the exact date and time at which events occur can be precisely recorded. Storing this information is the purpose of the time extension. Recording a timestamp for events is important, since this constitutes crucial information for many event log analysis techniques.

Name	*Prefix*	*URI*
Time	`time`	`http://www.xes-standard.org/time.xesext`

7.4.1 Attributes

Name	*Key*	*Components*	*Datatype*	*Description*
Time	`timestamp`	Event	`xs:dateTime`	The date and time at which the event occurred.

7.5 Semantic extension

Depending on the view on a process, type hierarchy artifacts may correspond to different concepts. For example, the name of an event (as specified by the Concept extension) may refer to the activity whose execution has triggered this event. However, this activity may be situated on a low level in the process meta-model, and be a part of higher-level, aggregate activities itself.

Besides events, also other elements of the XES type hierarchy may refer to a number of concepts at the same time (e.g., a log may refer to different process definitions, on different levels of abstractions). To express the fact, that one type artifact may represent a number of concepts in a process meta-model, the semantic extension has been defined.

It is assumed that there exists an ontology for the process meta-model, where every concept can be identified by a unique URI. The semantic extension defines an attribute, which allows to store a number of model references, as URIs, in any element of the XES type hierarchy.

Name	*Prefix*	*URI*
Semantic	`semantic`	`http://www.xes-standard.org/semantic.xesext`

7.5.1 Attributes

Name	*Key*	*Components*	*Datatype*	*Description*
Model reference	`modelReference`	Log Trace Event Meta	`xs:string`	References to model concepts in an ontology. Model References are stored in a literal string, as comma-separated URIs identifying the ontology concepts.

7.6 ID extension

The ID extension provides unique identifiers (UUIDs) for elements.

Name	*Prefix*	*URI*
Identity	`identity`	`http://www.xes-standard.org/identity.xesext`

7.6.1 Attributes

Name	*Key*	*Components*	*Datatype*	*Description*
Id	`id`	Log Trace Event Meta	ID	Unique identifier (UUID) for an element.

7.7 Cost extension

The cost extension defines a nested element to store information about the cost associated with activities within a log. The objective of this extension is to provide semantics to cost aspects that can be associated with events in a log. The definition associates three data elements with a particular cost element: the amount associated with the cost element as well as the cost driver that is responsible for incurring that cost and the cost type. As it is possible for more than one cost element to be associated with an event, the cost incurred per event is summarized using the `total` attribute. The `currency` element is also recorded once per event. Cost information can be recorded at the trace level (for instance, to be able to say that it costs $20 when a case is started). Cost information can also be recorded at the event level (for instance, for certain event types such as complete or canceled events) to capture the cost incurred in undertaking the activity by a resource.

Name	*Prefix*	*URI*
Cost	`cost`	`http://www.xes-standard.org/cost.xesext`

7.7.1 Attributes

Name	*Key*	*Components*	*Datatype*	*Description*
Total	`total`	Trace Event	`xs:double`	Total cost incurred for a trace or an event. The value represents the sum of all the cost amounts within the element.
Currency	`currency`	Trace Event	`xs:string`	The currency (using the ISO 4217:2008 Standard) of all costs of this element.
Drivers	`drivers`	Trace Event	List	A detailed list containing cost driver details.
Amount	`amount`	Meta	`xs:double`	The value contains the cost amount for a cost driver.
Driver	`driver`	Meta	`xs:string`	The value contains the id for the cost driver.
Type	`type`	Meta	`xs:string`	The value contains the cost type (e.g., `Fixed`, `Overhead`, `Materials`).

The `drivers` attribute shall contains any number of `driver` attributes, and every `driver` attribute shall contain the `amount` and `type` attribute, like follows:

```
<event>
  <string key="cost:currency" value="AUD" />
  <string key="cost:total" value="123.50" />
  <list key="cost:drivers">
    <values>
      <string key="driver" value="d2f4ee27">
        <float key="amount" value="21.40" />
        <string key="type" value="Labour" />
      </string>
      <string key="driver" value="abc124">
        <float key="amount" value="102.10" />
        <string key="type" value="Variable Overhead" />
      </string>
    </values>
  </list>
</event>
```

8. XES Conformance

Conformance to the XES part of this Standard is discussed in 8.1 and 8.2. In 8.1 and 8.2, *strictly conforming XES XML instance* and *conforming XES XML instance* refer to the metadata represented in the XES XML instance before any processing of the XES XML instance.

8.1 Strictly conforming XES instances

A strictly conforming XES instance shall consist solely of XES data elements as defined in Clause 4, and shall conform to the requirements of Clause 4. A strictly conforming XES XML instance

- Shall be a strictly conforming XES instance as defined before
- Shall conform to the requirements of Clause 5
- Shall not include XML elements or attributes that are not defined in Clause 5
- Shall not include mixed content

8.2 Conforming XES instances

A conforming XES instance may contain XES data elements as defined in Clause 4, and shall conform to the requirements of Clause 4. A conforming XES XML instance

- Shall be a conforming XES instance as defined before
- Shall conform to the requirements of Clause 5
- May include XML elements or attributes that are not defined in Clause 5
- May include mixed content

9. XESEXT Conformance

Conformance to the XESEXT part of this Standard is discussed in 9.1 and 9.2. In 9.1 and 9.2, *strictly conforming XESEXT XML instance* and *conforming XESEXT XML instance* refer to the metadata represented in the XESEXT XML instance before any processing of the XESEXT XML instance.

9.1 Strictly conforming XESEXT instances

A strictly conforming XESEXT instance shall consist solely of XESEXT data elements as defined in Clause 4.5, and shall conform to the requirements of Clause 4.5. A strictly conforming XESEXT XML instance

- Shall be a strictly conforming XES instance as defined before
- Shall conform to the requirements of Clause 6
- Shall not include XML elements or attributes that are not defined in Clause 6
- Shall not include mixed content

9.2 Conforming XESEXT instances

A conforming XESEXT instance may contain XESEXT data elements as defined in Clause 4.5, and shall conform to the requirements of Clause 4.5. A conforming XES XML instance

- Shall be a conforming XES instance as defined before
- Shall conform to the requirements of Clause 6
- May include XML elements or attributes that are not defined in Clause 6
- May include mixed content

Annex A

(informative)

History of XES

Unlike classical process analysis tools which are purely model-based (like simulation models), process mining requires event logs. Fortunately, today's systems provide detailed event logs. Process mining has emerged as a way to analyze systems (and their actual use) based on the event logs they produce [B1], [B2], [B3], [B4], [B6], [B25]. Note that, unlike classical data mining, the focus of process mining is on concurrent processes and not on static or mainly sequential structures. Also note that commercial Business Intelligence (BI for short) tools are not doing any process mining. They typically look at aggregate data seen from an external perspective (including frequencies, averages, utilization levels, and service levels). Unlike BI tools, process mining looks "inside the process" and allows for insights at a much more refined level.

The omnipresence of event logs is an important enabler of process mining, as analysis of run-time behavior is only possible if events are recorded. Fortunately, all kinds of information systems provide such logs, which include classical workflow management systems like FileNet and Staffware, ERP systems like SAP, case handling systems like BPM|one, PDM systems like Windchill, CRM systems like Microsoft Dynamics CRM, and hospital information systems like Chipsoft. These systems provide very detailed information about the activities that have been executed.

However, also all kinds of embedded systems increasingly log events. An embedded system is a special-purpose system in which the computer is completely encapsulated by or dedicated to the device or system it controls. Examples include medical systems like X-ray machines, mobile phones, car entertainment systems, production systems like wafer steppers, copiers, and sensor networks. Software plays an increasingly important role in such systems and, already today, many of these systems log events. An example is the "CUSTOMerCARE Remote Services Network" of Philips Medical Systems (PMS for short), which is a worldwide internet-based private network that links PMS equipment to remote service centers. Any event that occurs within an X-ray machine (like moving the table or setting the deflector) is recorded and can be analyzed remotely by PMS. The logging capabilities of the machines of PMS illustrate the way in which embedded systems produce event logs.

The MXML format [B10] has proven its use as a standard event log format in process mining. However, based on practical experiences with applying MXML in about one hundred organizations, several problems and limitations related to the MXML format have been discovered. One of the main problems is the semantics of additional attributes stored in the event log. In MXML, these are all treated as string values with a key and have no generally understood meaning. Another problem is the nomenclature used for different concepts. This is caused by MXML's assumption that strictly structured process would be stored in this format [B11].

To solve the problems encountered with MXML and to create a standard that could also be used to store event logs from many different information systems directly, a new event log format was developed. This new event log format is named XES, which stands for eXtensible Event Stream [B31]. This XES standard has been adopted for standardization by the IEEE Task Force Process Mining [B11].

Annex B

(informative)

Current status of XES

At the moment, there already exists a XES standard [B15] (which has not been endorsed by the IEEE), which comes with a reference implementation called OpenXES [B12]. The IEEE XES Standard differs from the XES 2.0 standard in the following respects (see also [B29]):

- Events in logs. In the XES 2.0 standard, every event had to be contained in some trace. In the IEEE XES Standard, events may also be contained by the log itself.

- Classifiers use ordered attribute keys. In the XES 2.0 standard, a classifier corresponds to a set of attribute keys. In the IEEE XES Standard, a classifier corresponds to a list of attribute keys.

- Trace classifier. In the XES 2.0 standard, only event classifiers were defined. In the IEEE XES Standard, also a trace classifier is defined, which allows to classify an entire trace. This can be useful in case the events in that trace lack the attributes for a corresponding event classifier.

- List attribute values. In the XES 2.0 standard, a list could not have any metadata as all attributes of the list were considered to be values of this list. In the IEEE XES Standard, there is a new element called `values` to hold the list values, which allows for the list metadata.

- Container attributes. In the XES 2.0 standard, a container attribute was defined. In the IEEE XES Standard, this attribute has been dropped. The list attribute suffices.

- Lifecycle extension. In the XES 2.0 standard, only the transition labels were defined. In the IEEE XES Standard, also the state labels (as introduced by [B21]) are used.

Annex C

(informative)

XES support

C.1 Tool support

The latest version of the XES standard, 2.0, is supported by the following tools:

- AProMore [B29], an advanced process model repository.
- Celonis Process Mining [B5], a process mining tool that retrieves and visualizes all of the process data saved in your IT systems.
- CoBeFra [B23], a comprehensive benchmarking framework for conformance checking.
- CoPrA Tool [B33], a tool for communication analysis and process mining of team processes.
- Disco [B11], a process mining tool that helps you to discover your processes.
- Flipflop [B22], a test and Flip net synthesis tool for the automated synthesis of surgical procedure models.
- JIRAVIEW [B18], a tool to extract data from JIRA through REST and create charts.
- Lean Document Production [B26] toolkit, a Xerox toolkit [B34] to model and analyze complex print operations.
- minit [B19], a modern process mining tool for complex process discovery, visualization and analysis.
- OpenXES [B12], The XES reference implementation, an Open Source Java library for reading, storing, and writing XES logs.
- PMLAB [B32], a scripting environment for Process Mining in Python, which allows to perform exploratory process-oriented computing and/or research in a process-oriented language.
- ProM 6 [B13], an extensible Process Mining framework, which is basically the breeding ground for many new process mining related techniques.
- RapidProM [B14], a ProM 6 Framework extension to RapidMiner 5.3.
- Rialto PI (Process Intelligence). The Rialto suite [B16] from Exeura covers Data Mining, Text Mining, Reasoning and Process Mining in one single environment.
- ruby-xes [B30], a Ruby library for generating XES event log.
- XES Python Tool [B25], a simple Python tool for generating XES files for Process Mining.
- XESame, a tool for extracting XES logs from databased, distributed with ProM 6.

- YAWL [B35], a BPM/Workflow system, based on a concise and powerful modelling language, that handles complex data transformations, and full integration with organizational resources and external Web Services.

C.2 Data support

A number of datasets have been published that use the XES standard:

- A real-life log from 5 Dutch Municipalities containing all building permit applications over a period of approximately five years [B8].
- A real-life log from a Dutch Financial Institute which contains some 262.200 events in 13.087 cases [B7].
- A real-life log from a Dutch Academic Hospital which contains some 150.000 events in over 1100 cases [B9].
- A real-life log from Volvo IT Belgium [B28]. The log contains events from an incident and problem management system called VINST.

These datasets are publicly available and can be used as benchmark datasets for many Computational Intelligence techniques. The use of the DOIs makes it easy to refer to the datasets.

C.3 Publication support

The XES standard has appeared in a number of publications:

- Aalst, W.M.P. van der, *Process Mining: Discovery, Conformance and Enhancement*. Heidelberg: Springer, 2011.
- Aalst, W.M.P. van der et al., “Process Mining Manifesto.” In *BPM 2011 Workshops*, LNBIP 99, edited by F. Daniel, et al., 169–194. Heidelberg: Springer, 2012.
- Aalst, W.M.P. van der and Dongen, B.F. van, “Discovering Petri Nets from Event Logs.” In *ToPNoC VII*, LNCS 7480, edited by K. Jensen et al., 372–422. Heidelberg: Springer, 2013.
- Accorsi, R., Wonnemann, C., and Dochow, S., “SWAT: A Security Analysis Toolkit for Reliably Process-aware Information Systems.” In *Workshop on Security Aspects of Process-aware Information*, 692–697. IEEE Computer Society, 2011.
- Accorsi, R., Stocker, T., and Müller, G., “On the Exploitation of Process Mining for Security Audits: The Process Discovery Case.” In *SAC 2012*, 1709–1716. ACM, 2012.
- Aruna Devi. C and Sudhamani, “Application Of Business Process Mining Using Control Flow Perspective In Manufacturing Unit,” *International Journal of Emerging Trends & Technology in Computer Science*, vol. 2, no. 5, pp. 130–133, 2013.
- Baumgrass, A., “Deriving Current-State RBAC Models from Event Logs.” In *International Workshop on Security Aspects of Process-aware Information Systems (SAPAIS), Proc. of the 6th International Conference on Availability, Reliability and Security (ARES)*, 667–672, IEEE Computer Society, 2011.

- Becker J. et al., "A Review of Event Formats as Enablers of event-driven BPM." In *BPM 2011 Workshops, Part I*, LNBIP 99, edited by F. Daniel et al., 433–445. Heidelberg: Springer, 2012.
- Bernardi, M.L. et al., "Using Discriminative Rule Mining to Discover Declarative Process Models with Non-atomic Activities." In *Rules on the Web. From Theory to Applications*, LNCS 8620, edited by A. Bikakis et al., 281–295, Heidelberg: Springer, 2014.
- Bose, R.P.J.C., Maggi, F.M., and Aalst, W.M.P. van der, "Enhancing Declare Maps Based on Event Correlations." In *BPM 2013*, LNCS 8094, edited by F. Daniel et al., 97–112. Heidelberg: Springer, 2013.
- Broucke, S. vanden et al., "A comprehensive benchmarking framework (CoBeFra) for conformance analysis between procedural process models and event logs in ProM." In *CIDM 2013, part of the IEEE Symposium Series on Computational Intelligence 2013*, SSCI 2013, 254–261. IEEE Computational Intelligence, 2013.
- Bui, D.B., Dazic, F., and Hecker, M., "Application of Tree-structured Data Mining for Analysis of Process Logs in XML format." In *AusDM 2012*, CR-PIT 134, edited by Y. Zhao et al., 109–118. Australian Computer Society, 2012.
- Burattin, A., *Process Mining Techniques in Business Environments: Theoretical Aspects, Algorithms, Techniques and Open Challenges in Process Mining*. LNBIP 207. Heidelberg: Springer, 2015.
- Burattin, A., Maggi, F.M., and Sperduti, A., "Conformance Checking Based on Multi-Perspective Declarative Process Models." arXiv:1503.04957, 2015.
- Caron, F. et al., "A Process Mining-based Investigation of Adverse Events in Care Processes." *Health Information Management Journal*, vol. 43, no. 1, pp. 16–25, 2014.
- Dolean, C.-C., "Mining Product Data Models: A Case Study." *Informatica Economică*, vol. 18, no. 1, pp. 69–82, 2014.
- Dongen, B.F. van, Shabani, Sh., "Relational XES: Data Management for Process Mining," Technical report, BPM Center Report BPM-15-02, BPMcenter.org, 2015.
- Engel, R. et al., "EDIminer: A Toolset for Process Mining from EDI Messages." In *CAiSE Forum 2013*, WS 998, edited by R. Deneckère and E. Proper, 146–153. CEUR-WS.org, 2013.
- Engel, R. et al., "Process Mining for Electronic Data Interchange." In *EC-Web 2011*, LNBIP 85, edited by C. Huemer and Th. Setzer, 77–88. Heidelberg: Springer, 2011.
- Gontar, B. and Gontar, Z., "Event Log Standards in BPM Domain." *Information Systems in Management*, vol. 1, no. 4, 293–304, 2012.
- Günther, C.W., "Process Mining in Flexible Environments." PhD thesis, Eindhoven University of Technology, 2009.
- Jones, W., "The Future of Personal Information Management, Part 1: Our Information, Always and Forever." *Synthesis Lectures on Information Concepts, Retrieval, and Services*, vol. 4, no. 1, 1–125, 2012.

- Kaes, G. et al., “Flexibility Requirements in Real-World Process Scenarios and Prototypical Realization in the Care Domain.” In *OTM 2014 Workshops*, LNCS 8842, edited by R. Meersman et al., 55–64. Heidelberg: Springer, 2014.

- Khoadandelou, G. et al., “Process Mining Versus Intention Mining.” In *EMMSAD 2013*, LNBIP 147, edited by S. Nurcan et al., 466–480. Heidelberg: Springer, 2013.

- Krinkin, K., Kalishenko, E., and Prakash, S.P.S., “Process Mining Approach for Traffic Analysis in Wireless Mesh Networks.” In *NEW2AN/ruSMART 2012*, LNCS 7469, edited by S. Andreev et al., 260–269. Heidelberg: Springer, 2012.

- Le, N.T.T. et al., “Representing, Simulating and Analysing Ho Chi Minh City Tsunami Plan by Means of Process Models.” arXiv:1312.4851v2, 2013.

- Ly, L.T. et al., “Data Transformation and Semantic Log Purging for Process Mining.” In *CAiSE 2012*, LNCS 7328, edited by J. Ralyté, J. et al., 238–253. Heidelberg: Springer, 2012.

- Mans, R.S., Aalst, W.M.P. van der, and Verwersch, R.J.B., *Process Mining in Healthcare: Evaluating and Exploiting Operational Healthcare Processes*. Heidelberg: Springer, 2015.

- Mans, R.S. et al., “Process Mining in Healthcare: Data Challenges When Answering Frequently Posed Questions.” In *ProHealth 2012/KR4HC 2012*, LNAI 7738, edited by R. Lenz et al., 140–153. Heidelberg: Springer, 2013.

- Muehlen, M. zur and Swenson, K.D., “BPAF: A Standard for the Interchange of Process Analytics Data.” In *BPM 2010 Workshops*, LNBIP 66, 170–181. Heidelberg: Springer, 2011.

- Mueller-Wickop, N. and Schultz, M., “ERP Event Log Preprocessing: Timestamps vs. Accounting Logic.” In *DESRIST 2013*, LNCS 7939, edited by J. vom Brocke et al., 105–119. Heidelberg: Springer, 2013.

- Nammakhunt, A. et al., “Process Mining: Converting from MS-Access Database to MXML Format.” In *2012 Tenth International Conference on ICT and Knowledge Engineering*, 205–212. IEEE Computer Society, 2012.

- Redlich, D. et al., “Introducing a Framework for Scalable Dynamic Process Discovery.” In *EEWC 2014*, LNBIP 174, edited by D. Averio et al., 151–166. Heidelberg: Springer, 2014.

- Sahlabadi, M., Muniyandi, and R.C., Shukur, Z., “Detecting Abnormal Behavior in Social Network Websites by using a Process Mining Technique.” *Journal of Computer Science*, vol. 10, no. 3, pp. 393–420, 2014.

- Seeber, I., Maier, R., and Weber, B., “CoPrA: A Process Analysis Technique to Investigate Collaboration in Groups.” In *HICSS 2012*, 362–372. IEEE Computer Society, 2012.

- Sellami, R, Gaaloul, W., and Moalla, S., “An Ontology for Workflow Organizational Model Mining.” In *WETICE 2012*, edited by S. Reddy and K. Drira, 199–204. IEEE Computer Society, 2013.

- Singh, L. and Chetty, G., “A Comparative Study of MRI Data Using Various Machine Learning and Pattern Recognition Algorithms to Detect Brain Abnormalities.” In *AusDM 2012*, CR-PIT 134, edited by Y. Zhao et al., 157–165. Australian Computer Society, 2012.

- Suriadi, S. et al., "Root Cause Analysis with Enriched Process Logs." In *BPM 2012 Workshops*, LNBIP 132, edited by M. La Rosa and P. Soffer, 174–186. Heidelberg: Springer, 2012.

- Verbeek, H.M.W. et al., "XES, XESame, and ProM 6." In *CAiSE Forum 2010*, LNBIP 72, edited by P. Soffer and E. Proper, 60–75, 2010. Heidelberg: Springer, 2010.

- Wakup, C. and Desel, J., "Analyzing a TCP/IP-Protocol with Process Mining Techniques." In *BPM 2014 Workshops*, LNBIP 202, edited by F. Fournier and J. Mendling, 353–364. Heidelberg: Springer, 2015.

- Werf, J.M.E.M. van der and Verbeek, H.M.W., "Online Compliance Monitoring of Service Landscapes." In *BPM 2014 Workshops*, LNBIP 202, edited by F. Fournier and J. Mendling, 89–95. Heidelberg: Springer, 2015.

- Vasilecas, O., Svickas, T., and Lebedys, E., "Directed Acyclic Graph Extraction from Event Logs." In *ICIST 2014*, CCIS 465, edited by G. Dregvaite and R. Damasevicius, 172–181. Heidelberg: Springer, 2014.

- Vera-Baquero, A. et al., "Business process improvement by means of Big Data based Decision Support Systems: a case study on Call Centers." *International Journal of Information Systems and Project Management*, vol. 3, no. 1, 5–26, 2015.

- Weber, Ph., "A Framework for the Analysis and Comparison of Process Mining Algorithms." PhD thesis, University of Birmingham, 2013.

- Westergaard, M. van Dongen, B.F., "KeyValuesets: Event Logs Revisited, Technical report," BPM Center Report, BPM-13-25, BPMcenter.org, 2013.

- Wynn, M.T. et al., "A Framework for Cost-Aware Process Management: Cost Reporting and Cost Prediction." *Journal of Universal Computer Science*, vol. 20, no. 3, pp. 406–430, 2014.

Annex D

(informative)

XESEXT Example

We have chosen the Semantic Extension (see 7.5) to exemplify the XESEXT format for XES extensions. This extension defines, on each level of abstraction (log, trace, event, and meta), the same string-based attribute `modelReference`. Attributes can be defined on all four levels of abstraction, similar to attribute declarations in XES (while omitting the value attribute). For every defined attribute, the XESEXT document may feature an arbitrary number of alias mappings as child elements. These mappings define a human-readable alias for the attribute within a given namespace (typically a country code, used for localization).

```
<?xml version="1.0" encoding="UTF-8" ?>
<xesextension name="Semantic"
              prefix="semantic"
              uri="http://code.fluxicon.com/xes/semantic.xesext">

  <log>
    <string key="modelReference">
      <alias mapping="EN" name="Ontology Model Reference" >
      <alias mapping="DE" name="Ontologie-Modellreferenz" />
      <alias mapping="FR" name="Référence au Modèle Ontologique" />
      <alias mapping="ES" name="Referencia de Modelo Ontológico" />
      <alias mapping="PT" name="Referência de Modelo Ontológico" />
    </string>
  </log>

  <trace>
    <string key="modelReference">
      <alias mapping="EN" name="Ontology Model Reference" />
      <alias mapping="DE" name="Ontologie-Modellreferenz" />
      <alias mapping="FR" name="Référence au Modèle Ontologique" />
      <alias mapping="ES" name="Referencia de Modelo Ontológico" />
      <alias mapping="PT" name="Referência de Modelo Ontológico" />
    </string>
  </trace>

  <event>
    <string key="modelReference">
      <alias mapping="EN" name="Ontology Model Reference" />
      <alias mapping="DE" name="Ontologie-Modellreferenz" />
      <alias mapping="FR" name="Référence au Modèle Ontologique" />
      <alias mapping="ES" name="Referencia de Modelo Ontológico" />
      <alias mapping="PT" name="Referência de Modelo Ontológico" />
    </string>
  </event>

  <meta>
    <string key="modelReference">
      <alias mapping="EN" name="Ontology Model Reference" />
      <alias mapping="DE" name="Ontologie-Modellreferenz" />
      <alias mapping="FR" name="Référence au Modèle Ontologique" />
```

```
      <alias mapping="ES" name="Referencia de Modelo Ontológico" />
      <alias mapping="PT" name="Referência de Modelo Ontológico" />
    </string>
  </meta>

</xesextension>
```

Annex E

(informative)

XES Schema definition (XSD)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">

  <xs:element name="log" type="LogType"/>

  <!-- Attributables -->
  <xs:complexType name="AttributableType">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="string" minOccurs="0" maxOccurs="unbounded"
                  type="AttributeStringType" />
      <xs:element name="date" minOccurs="0" maxOccurs="unbounded"
                  type="AttributeDateType" />
      <xs:element name="int" minOccurs="0" maxOccurs="unbounded"
                  type="AttributeIntType" />
      <xs:element name="float" minOccurs="0" maxOccurs="unbounded"
                  type="AttributeFloatType" />
      <xs:element name="boolean" minOccurs="0" maxOccurs="unbounded"
                  type="AttributeBooleanType" />
      <xs:element name="id" minOccurs="0" maxOccurs="unbounded"
                  type="AttributeIDType" />
      <xs:element name="list" minOccurs="0" maxOccurs="unbounded"
                  type="AttributeListType" />
    </xs:choice>
  </xs:complexType>

  <!-- String attribute -->
  <xs:complexType name="AttributeStringType">
    <xs:complexContent>
      <xs:extension base="AttributeType">
        <xs:attribute name="value" use="required" type="xs:string" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Date attribute -->
  <xs:complexType name="AttributeDateType">
    <xs:complexContent>
      <xs:extension base="AttributeType">
        <xs:attribute name="value" use="required" type="xs:dateTime" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Integer attribute -->
  <xs:complexType name="AttributeIntType">
    <xs:complexContent>
      <xs:extension base="AttributeType">
```

```
        <xs:attribute name="value" use="required" type="xs:long" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Floating-point attribute -->
  <xs:complexType name="AttributeFloatType">
    <xs:complexContent>
      <xs:extension base="AttributeType">
        <xs:attribute name="value" use="required" type="xs:double" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Boolean attribute -->
  <xs:complexType name="AttributeBooleanType">
    <xs:complexContent>
      <xs:extension base="AttributeType">
        <xs:attribute name="value" use="required" type="xs:boolean" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- ID attribute -->
  <xs:complexType name="AttributeIDType">
    <xs:complexContent>
      <xs:extension base="AttributeType">
        <xs:attribute name="value" use="required" type="xs:string" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- List attribute -->
  <xs:complexType name="AttributeListType">
    <xs:complexContent>
      <xs:extension base="AttributeType">
        <xs:sequence>
          <xs:element name="values" minOccurs="1" maxOccurs="1"
                      type="AttributeType" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Extension definition -->
  <xs:complexType name="ExtensionType">
    <xs:attribute name="name" use="required" type="xs:NCName" />
    <xs:attribute name="prefix" use="required" type="xs:NCName" />
    <xs:attribute name="uri" use="required" type="xs:anyURI" />
  </xs:complexType>

  <!-- Globals definition -->
  <xs:complexType name="GlobalsType">
    <xs:complexContent>
      <xs:extension base="AttributableType">
        <xs:attribute name="scope" type="xs:NCName" use="required" />
      </xs:extension>
```

```
    </xs:complexContent>
  </xs:complexType>

  <!-- Classifier definition -->
  <xs:complexType name="ClassifierType">
    <xs:attribute name="name" type="xs:NCName" use="required" />
    <xs:attribute name="scope" type="xs:NCName" use="required" />
    <xs:attribute name="keys" type="xs:token" use="required" />
  </xs:complexType>

  <!-- Attribute -->
  <xs:complexType name="AttributeType">
    <xs:complexContent>
      <xs:extension base="AttributableType">
        <xs:attribute name="key" use="required" type="xs:Name" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Elements may contain attributes -->
  <xs:complexType name="ComponentType">
    <xs:complexContent>
      <xs:extension base="AttributableType" />
    </xs:complexContent>
  </xs:complexType>

  <!-- Logs are elements that may contain traces -->
  <xs:complexType name="LogType">
    <xs:complexContent>
      <xs:extension base="ComponentType">
        <xs:sequence>
          <xs:element name="extension" minOccurs="0"
                      maxOccurs="unbounded" type="ExtensionType" />
          <xs:element name="global" minOccurs="0"
                      maxOccurs="2" type="GlobalsType" />
          <xs:element name="classifier" minOccurs="0"
                      maxOccurs="unbounded" type="ClassifierType" />
          <xs:element name="trace" minOccurs="0" maxOccurs="unbounded"
                      type="TraceType" />
          <xs:element name="event" minOccurs="0" maxOccurs="unbounded"
                      type="EventType" />
        </xs:sequence>
        <xs:attribute name="xes.version" type="xs:decimal"
                      use="required" />
        <xs:attribute name="xes.features" type="xs:token" />
        <xs:attribute name="openxes.version" type="xs:string" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Traces are elements that may contain events -->
  <xs:complexType name="TraceType">
    <xs:complexContent>
      <xs:extension base="ComponentType">
        <xs:sequence>
          <xs:element name="event" minOccurs="0" maxOccurs="unbounded"
                      type="EventType"/>
```

```
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Events are elements -->
  <xs:complexType name="EventType">
    <xs:complexContent>
      <xs:extension base="ComponentType">
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

</xs:schema>
```

Annex F

(informative)

XESEXT Schema definition (XSD)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">

  <xs:element name="xesextension">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="log" type="AttributableType" minOccurs="0" />
        <xs:element name="trace" type="AttributableType"
                    minOccurs="0" />
        <xs:element name="event" type="AttributableType"
                    minOccurs="0" />
        <xs:element name="meta" type="AttributableType"
                    minOccurs="0" />
      </xs:sequence>
      <xs:attribute name="name" type="xs:NCName" use="required" />
      <xs:attribute name="prefix" type="xs:NCName" use="required" />
      <xs:attribute name="uri" type="xs:anyURI" use="required" />
    </xs:complexType>
  </xs:element>

  <!-- Attributes -->
  <xs:complexType name="AttributableType">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="string" type="AttributeType" />
      <xs:element name="date" type="AttributeType" />
      <xs:element name="int" type="AttributeType" />
      <xs:element name="float" type="AttributeType" />
      <xs:element name="boolean" type="AttributeType" />
      <xs:element name="id" type="AttributeType" />
      <xs:element name="list" type="AttributeType" />
    </xs:choice>
  </xs:complexType>

  <!-- Attribute -->
  <xs:complexType name="AttributeType">
    <xs:sequence>
      <xs:element name="alias" type="AliasType"
                  minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="key" type="xs:Name" use="required" />
  </xs:complexType>

  <!--  Alias definition, defining a mapping alias for an attribute -->
  <xs:complexType name="AliasType">
    <xs:attribute name="mapping" type="xs:NCName" use="required" />
    <xs:attribute name="name" type="xs:string" use="required" />
  </xs:complexType>
```

```
</xs:schema>
```

Annex G

(informative)

Bibliography

Bibliographical references are resources that provide additional or helpful material but do not need to be understood or used to implement this standard. Reference to these resources is made for informational use only.

[B1] Aalst, W.M.P. van der et al., “Business Process Mining: An Industrial Application.” *Information Systems*, vol. 32, no. 5, pp. 713–732, 2007.

[B2] Aalst, W.M.P. van der et al., “Workflow Mining: A Survey of Issues and Approaches.” *Data and Knowledge Engineering*, vol. 47, no. 2, pp. 237–267, 2003.

[B3] Aalst, W.M.P. van der, Weijters, A.J.M.M., and Maruster, L., “Workflow Mining: Discovering Process Models from Event Logs.” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128–1142, 2004

[B4] Agrawal, R., Gunopulos, D., and Leymann, F., “Mining Process Models from Workflow Logs.” In *EDBT 1998*, LNCS 1377, edited by H.-J. Schek et al., 469–483. Heidelberg: Springer, 1998.

[B5] Celonis, “Celonis Process Mining.” Accessed July 10, 2015, http://www.celonis.de/en/.

[B6] Datta, A., “Automating the Discovery of As-Is Business Process Models: Probabilistic and Algorithmic Approaches.” *Information Systems Research*, vol. 9, no. 3, pp. 275–301, 1998.

[B7] Dongen, B.F. van, “BPI Challenge 2012.” Accessed June 1, 2015, http://dx.doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f.

[B8] Dongen, B.F. van, “BPI Challenge 2015.” Accessed June 1, 2015, http://dx.doi.org/10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1.

[B9] Dongen, B.F. van, “Real-life event logs - Hospital log.” Accessed June 1, 2015, http://dx.doi.org/10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54.

[B10] Dongen, B.F. van and Aalst, W.M.P. van der, “A Meta Model for Process Mining Data.” In *CAiSE 2005 Workshops (EMOI-INTEROP Workshop), vol. 2*, edited by J. Casto and E. Teniente, 309–320. Porto: FEUP, 2005.

[B11] Eindhoven University of Technology, “IEEE CIS Task Force on Process Mining.” Accessed June 2, 2015, http://www.win.tue.nl/ieeetfpm.

[B12] Eindhoven University of Technology. “OpenXES.” Accessed June 1, 2015, http://www.openxes.org.

[B13] Eindhoven University of Technology, “ProM 6.” Accessed June 1, 2015, http://www.promtools.org/prom6.

[B14] Eindhoven University of Technology, “RapidProM.” Accessed June 1, 2015, http://rapidprom.org/.

[B15] Eindhoven University of Technology, “XES Standard.” Accessed June 1, 2015, http://www.xes-standard.org.

[B16] Exeura, “Rialto.” Accessed June 30, 2015, http://www.exeura.eu/en/products/rialto.

[B17] Fluxicon. “Disco.” Accessed June 1, 2015, http://fluxicon.com/disco.

[B18] GitHub. “JIRAVIEW.” Accessed June 2, 2015, https://github.com/godatadriven/jiraview.

[B19] GRADIENT ECM. “minit.” Accessed October 5, 2015, http://minitlabs.com.

[B20] Günther, C.W., "Process Mining in Flexible Environments." PhD thesis, Eindhoven University of Technology, 2009.

[B21] IEEE Computational Intelligence Society, "The eXtensible Event Stream (XES) standard," XES Webinar, IEEE CIS Video Library. Accessed June 2, 2015, http://cis.ieee.org/cis-educational-repository/cis-video-collection/cis-video-library.html?vid=102289302.

[B22] Irisa Rennes. "Flipflop." Accessed June 2, 2015, http://tinyurl.com/oql6f3y (last accessed June 2015)

[B23] KU Leuven. "CoBeFra." Accessed June 2, 2015, http://processmining.be/cobefra/

[B24] Muehlen, M. zur and Swenson, K.D., "BPAF: A Standard for the Interchange of Process Analytics Data. In *BPM 2010 Workshops*, LNBIP 66, 170–181. Heidelberg: Springer, 2011.

[B25] Python Software Foundation, "XES Python Tool." Accessed June 2, 2015, https://pypi.python.org/pypi/xes/.

[B26] Rai, S. et al., "LDP Lean Document Production—O.R.-Enhanced Productivity Improvements for the Printing Industry." *Interfaces*, vol. 39, no. 1, pp. 69–90, 2009. http://dx.doi.org/10.1287/inte.1080.0413.

[B27] Rozinat, A. and Aalst, W.M.P. van der, "Conformance Checking of Processes Based on Monitoring Real Behavior." *Information Systems*, vol. 33, no. 1, pp. 64–95, 2008.

[B28] Steeman, W., "BPI Challenge 2013, incidents." Accessed June 2, 2015, http://dx.doi.org/10.4121/uuid:500573e6-accc-4b0c-9576-aa5468b10cee.

[B29] The AProMore Initiative. "AProMore." Accessed June 2, 2015, http://apromore.org/.

[B30] The Ruby Toolbox. "ruby-xes." Accessed June 2, 2015, https://www.ruby-toolbox.com/projects/ruby-xes.

[B31] Verbeek, H.M.W. et al., "XES, XESame, and ProM 6." In *CAiSE Forum 2010*, LNBIP 72, edited by P. Soffer and E. Proper, 60–75, 2010. Heidelberg: Springer, 2010.

[B32] Universitat Politècnica de Catalunya, "PMLAB", http://www.cs.upc.edu/~jcarmona/PMLAB/.

[B33] University of Innsbruck and Università degli Studi di Milano. "CoPrA Tool." Accessed June 2, 2015, http://www.copra-tool.eu/.

[B34] Xerox Research Center Webster, "Lean Document Production (LDP) Earns Top Honors." Accessed June 30, 2015, http://www.xrcw.xerox.com/About-XRCW/News/Lean-Document-Production-LDP-Earns-Top-Honors.

[B35] YAWL Foundation, "YAWL." Accessed June 2, 2015, http://yawlfoundation.org/.

Annex R

(informative)

Revision history

This annex contains the revision history of this Standards proposal, and is to be omitted from the final Standard proposal.

R.1 May 21st, 2015: Initial revision

R.2 June 1st, 2015: Created annex for tool, data, and publication support

- Fixed some typos.

R.3 June 2nd, 2015: Fixed citation style to Chicago.

- Resorted citations accordingly.
- Fixed references to citations.

R.4 June 26th, 2015: Comments of Moe Wynn, fixed some other glitches.

- Textual changes based on comments of Moe.
- Fixed event notation in Clause 4.4.3.
- Fixed undefined references in Clause 8 and Clause 9.

R.5 June 29th, 2015: Comments of Lijie Wen.

- Textual changes based on comments of Lijie.

R.6 June 30th, 2015: Comments of JC Bose and Walter van Herle.

- Added Xerox Lean Document Production toolkit, as suggested by JC Bose, together with two references.
- Added Rialto PI, as suggested by Walter van Herle, together with a reference.

R.7 July 10th, 2015: Comments of Alexander Rinke.

- Added Celonis Process Mining tool, as suggested by Alexander Rinke.

- Updated all fields.

R.8 October 2nd, 2015: Comments of Josep Carmona.

- Added PMLAB as tool that supports XES, with a reference.
- Added all members to the WG.
- Updated month in header.

R.9 October 6th, 2015: Comments of Michal Rosik.

- Added minit as tool that supports XES, with reference.
- Added Michal as WG participant.

R.10 November 4th, 2015: Comments of Zbigniew Paszkiewicz.

- Minor textual changes.

R.11 December 2nd, 2015: First draft for balloting

- Updated header