The XML Bookmark Exchange Language

Release 1.0

Fred L. Drake, Jr.

28 October, 1998

Corporation for National Research Initiatives (CNRI)
1895 Preston White Drive, Reston, Va 20191, USA
E-mail: fdrake@acm.org

Abstract

The XML Bookmark Exchange Language (XBEL) is a rich interchange format for "bookmark" data as used by most Internet browsers. This document describes the origin of the design, the requirements which drove the design process, and the resulting document type.

Contents

1 Introduction

The XML Bookmark Exchange Language, or XBEL, is an interchange format for the hierarchical bookmark data used by current Internet browsers. It is defined as an application of the Extensible Markup Language, or XML [BPSM98].

This section descibes the origin of the effort which created the XML Bookmark Exchange Language (XBEL), identifies the contributors, and provides information on the availability of the public text of the DTD and additional documentation on the applications which support XBEL.

1.1 Origins

The XML Bookmark Exchange Language is a product of the Python XML Special Interest Group (XML-SIG), sponsored by the Python Software Activity (PSA). The initial intent of the XBEL effort was to create a demonstration of XML facilities available to Python programmers which would also be useful.

1.2 Contributors

The initial idea for XBEL was contributed by Mark Hammond. Mark sent his idea to the Python XML-SIG mailing list. This was closely followed by discussions and additional ideas by many of the list participants. The following people contributed to the design of the DTD and the related software (listed in alphabetical order by last name):

Fred L. Drake, Jr. (fdrake@acm.org)
Documentation. Design input on DTD and the storage of metadata. Implemented direct support for XBEL in Grail.

Stefane Fermigier (fermigie@math.jussieu.fr)
Modified implementation of software for Internet Explorer Favorites conversion using his original Python DOM implementation.

Lars Marius Garshol (larsga@ifi.uio.no)
Extended the concept to cover all Internet browsers bookmarks and came up with the name and acronym. Implemented support for Navigator and Opera bookmark formats.

Geir Ove Gronmo (grove@infotek.no)
General input on XML and the desired level of complexity.

Marc van Grootel (bwaumg@urc.tue.nl)
Design input on the DTD, storage of metadata, and comments on the use of XBEL with architectural forms.

Mark Hammond (MHammond@skippinet.com.au)
Original concept and DTD for an archival storage format for Internet Explorer "Favorites."

Jack Jansen (Jack.Jansen@cwi.nl)
General input on potential advanced applications.

Andrew M. Kuchling (akuchling@acm.org)
Implemented conversion software between XBEL and Lynx bookmarks.

Fredrik Lundh (fredrik@pythonware.com)
Initial software implementation for Internet Explorer.

Sean McGrath (digitome@iol.ie)
General input on XML and document type definitions.

Greg Stein (gstein@lyra.org)
General input on XML Namespaces and moderator of complexity.

Walter R. Underwood (wunder@infoseek.com)
General input on the use of XML character entites instead of adding general entities, and discussion on date/time values in XML.

1.3 Availability

Information on XBEL, including the public text and this document, is available on the PSA Web site at http://www.python.org/topics/xml/xbel/ [Dra]. Please refer to this Web resource for information on new versions, DTD variants, and supporting software.

The public text for XBEL will be made available through a SOCAT catalog at available at: http://www.python.org/topics/xml/dtds/catalog. This catalog may be used by including a DELEGATE entry in a catalog already used by XML processing software. The DELEGATE entry should be:

```
DELEGATE "+//IDN python.org" "http://www.python.org/topics/xml/dtds/catalog"
```

1.4 Formal Identification

The XBEL DTD documented in this report has the Formal Public Identifier:

```
+//IDN python.org//DTD XML Bookmark Exchange Language 1.0//EN//XML
```

Valid instances of this document type may use the following document type declaration:

```
<!DOCTYPE xbel
  PUBLIC "+//IDN python.org//DTD XML Bookmark Exchange Language 1.0//EN//XML"
         "http://www.python.org/topics/xml/dtds/xbel-1.0.dtd">
```

2 Requirements

This section describes the functional capabilities which this document type supports. There are three categories of functionality supported: basic bookmark exchange between browsers, data storage for advanced Internet resource management tools, and simplicity in extending the DTD if needed for specific applications.

2.1 Relation to Browser Functionality

XBEL instances must be able to describe sufficient data to represent the bookmarks of all major Internet browsers. It must be possible to convert browser-specific bookmark data to XBEL in a lossless manner, though specific conversions may remove data for application-specific reasons. It is especially important to consider privacy issues when exchanging bookmark data.

Conversion from XBEL to a browser-specific format may lose information when the data originates from a browser supporting bookmark features not available in all browsers. It is expected that software implementing the conversion be able to warn the user if conversion will cause the loss of information, as appropriate.

2.2 Advanced Application Support

XBEL must be able to support interchange requirements for applications not currently implemented as part of typical Internet browsers, including (but not limited to!) application-specific preference and history information which only pertains to specific bookmarks, metadata information, and alternate sources or formats for the documents.

It must be possible for applications to operate on subsets of the information stored in an XBEL instance without affecting private information stored by other applications. Application-specific data stored in an XBEL instance may be simple text or may be heavily structured.

2.3 Extensibility

Some ability to extend the document type definition is required to encourage reuse of the existing design. Due to the use of XML, only a minimum of inherant flexibility is required, as new document types may be formed using namespaces or by allowing the use of well-formed but possibly invalid markup [BHL98].

3 XBEL Document Structure

This section describes the structure of XBEL documents. This includes information on each element and attribute defined in the DTD. Some descriptions include references to the parameter entities used to construct the DTD; these are described in Section 4.1, "Use of Parameter Entities."

3.1 Date/time Attribute Values

Several attributes defined in this document type require date/time values stored as CDATA. For these attributes, the value must be formatted as an ISO 8601:1988 value containing a date [fS88, Hou93, Kuh98]. A time value should be supplied whenever the information is available to the application which set the value. The format of the values is restricted to the forms specified in the profile defined in *Date and Time Formats* [WW98]. Attributes which require this form of value are described below as having a *date/time value* rather than a CDATA value.

3.2 Top-level Information

This section describes the top-level element type of XBEL documents.

The `<xbel>` Element

The `<xbel>` element defines the top-level data structure stored in an XBEL instance. It may contain optional `<title>`, `<info>`, and `<desc>` elements, followed by any number of elements from `%nodes.mix;`. This is similar to the `<folder>` element, but it may not be nested and carries different attributes.

Attributes The `<xbel>` element carries a `version` attribute which has a fixed value that specifies the version of the XBEL DTD. Other attributes indicate the similarity to the `<folder>` element.

`version`, ***fixed***
: Fixed value which specifies the version of the DTD in use.

`id`
: ID value to allow linking to this element; only the `<alias>` element's `ref` attribute supports a corresponding IDREF value.

`added`
: Date/time value which can be used to record when the collection of bookmarks was created.

Processing Expectations The `<xbel>` element is in many ways similar to a `<folder>` element, but may not be "folded." Auxillary information, such as an optional `<title>` element, may be shown in a substantially different way than for `<folder>` in a user interface.

3.3 Common Elements

Elements described in this section may occur in different contexts within an XBEL instance, but share fundamental semantic interpretation in each case.

The `<title>` Element

The `<title>` element is used to mark the title associated with the immediately enclosing element. It is used for the `<xbel>`, `<folder>`, and `<bookmark>` elements. This element is always optional and may contain only character data.

Processing Expectations Software which presents bookmark information to the user in any form should use the content of this element to identify the resource to the user. Additional information may be needed to make the identification unambiguous.

Applications may use the text of the `<title>` during search operations.

Rationale Many Internet resources are described by a short title, often displayed by the bookmarking facilities. Storing the title allows a significant improvement in user interface responsiveness when compared to retrieving the resource to reload the title. Title storage is the approach taken by all browsers known to the XBEL designers.

The `<desc>` Element

The `<desc>` element is used to store a human-readable description of the enclosing element. For a `<folder>` or the `<xbel>` element, this may be used to more thoroughly explain the subject of the bookmarks stored in the collection and why they may be interesting. For a `<bookmark>`, a summary of the resource pointed to by the bookmark may be more appropriate. This element is always optional and may contain only character data.

Processing Expectations

The content of this element may be displayed to a user requesting more information on the folder or bookmark containing the description. In the case of a `<bookmark>`, this can be used before actually making a request over the network to retrieve the resource.

Applications may use the text of the `<desc>` during search operations.

Rationale

Many Internet browsers support simple annotation of bookmark data with human readable text. This element is required to support exchange of this data.

The `<info>` Element

The `<info>` element is used to store metadata related to the immediately enclosing element. The intended use is for `<info>` to store a series of `<metadata>` elements, each of which "belongs" to some application. An "application" in this sense may be either a program, such as a specific Internet browser, or a more general metadata scheme, such as the Dublin Core [Dub97].

The `<info>` element is always optional. If present, it must contain one or more `<metadata>` elements.

Processing Expectations

Applications are expected to ignore `<info>` elements if they are not able to deal with the contents of constituent `<metadata>` elements. Whether or not `<info>` elements should be "passed through" transparently or removed depends on the purpose of the processing application, but an effort should be made to retain the information whenever the enclosing element is retained, even in a modified form.

Rationale

This element provides a clean way of isolating application-specific metadata from more generally supported constructs within the bookmark data.

The `<metadata>` Element

The `<metadata>` element is used as a container for all auxillary information related to a node which belongs to a single metadata scheme. The specific contents of `<metadata>` is highly dependent on the metadata scheme which applies; XML namespaces should be used to identify explicit markup used within the element.

`<metadata>` elements are always optional. Note that an `<info>` element which contains no `<metadata>` elements must be removed.

Attributes

`owner`, ***required***
: CDATA value specifies the application which "owns" the content of the element. The value of this attribute should be a URI which refers to a definition of the application and content structure in either human- or machine-processible form. It is not required that the URI be addressable through the network.

It is expected that namespace attributes will be added to the element to specify the markup defined for the content of the `<metadata>` element.

Processing Expectations Within an `<info>` element, each `<metadata>` element is required to have a unique value for the `owner` attribute. Programs which modify the contents of `<metadata>` elements should ensure that only one `<metadata>` exists for any `owner` value normally modified by the application within affected `<info>` elements. `<metadata>` elements for other owners should remain unaffected.

Specific interpretation of `<metadata>` content is highly dependent on both the `owner` and the application, and is not otherwise within the scope of this document.

Rationale The `<metadata>` element is required to support owner identification. It is entirely reasonable for multiple owners of data to share a document type for their information, but otherwise require separate processing. The Resource Description Framework provides an example of an approach which would require multiple applications to share a namespace [LS98, BGL98]. Some additional form of ownership identification is required to ensure processors can avoid destroying each other's data.

3.4 Data Organization

The elements described in this section are used to impose organization on a collection of `<bookmark>` nodes. `<folder>` is used to support hierarchical organization and `<separator>` is used to support non-hierarchical organization.

The `<folder>` Element

The `<folder>` element is the element used to support hierarchical data organization. It is the only element type which is allowed to nest within itself.

This element may contain optional `<title>`, `<info>` and `<desc>` elements. After this, any number of elements from `%nodes.mix;` are allowed.

Attributes

`id`
: ID value to allow linking to this element; only the `<alias>` element's `ref` attribute supports a corresponding IDREF value.

`added`
: Date/time value which records when the folder was added to the bookmark collection represented by the instance.

`folded`
: Token which records whether the contents of the folder should be displayed by default in a user interface. The value may be `yes` or `no`.

Processing Expectations User interfaces should display `<folder>` elements as collapsing lists, allowing the user to display or hide the contents of the element on demand. Appropriate behavior outside of user interfaces is expected to be application specific.

Rationale The `<folder>` element may be used to represent hierarchical relationships within a collection of bookmarks, as deployed in current Internet browsers.

The `<separator>` Element

The `<separator>` element can be used to separate bookmarks within a collection in a non-hierarchical fashion. It may be used within a `<folder>` or the `<xbel>` element.

Processing Expectations The presence of this element may be represented by displaying a horizontal line or vertical whitespace in an interactive user interface or printed representation.

Rationale A simple separator is required to support the bookmark structures of existing Internet browsers.

3.5 Bookmark Data

Only one element type is used to encapsulate information specific to an individual bookmark. No need for alternate elements has been demonstrated.

The `<bookmark>` Element

A `<bookmark>` element is used to store information about a specific resource. This element may contain the optional elements `<title>`, `<info>` and `<desc>`.

Attributes The `<bookmark>` element carries more attributes than other elements defined in XBEL. These attributes are used to carry much of the common information maintained on bookmarks by the major browsers.

`href`, ***required***
URI which specifies the resource described by the `<bookmark>` element.

`id`
ID value to allow linking to this element; only the `<alias>` element's `ref` attribute supports a corresponding IDREF value.

`added`
Date/time value which indicates when the `<bookmark>` element was added to the bookmark collection.

`modified`
Date/time value which records the time of the last known change to the resource identified by the `<bookmark>`.

`visited`
Date/time value which represents the time of the user's last "visit" to the resource. Note that the value for `modified` may be more recent than the value for `visited` if software is used that checks for resources which have changed since the user last visited the resource. This feature is increasingly common in browsers.

Processing Expectations In a user interface, `<bookmark>` should typically be represented by the contents of the `<title>` element, if present. The representation of the bookmark should be "hot," allowing traversal to the referenced resource by the user. Additional information on the resource, such as the description given in a `<desc>` element, should be available to the user on demand.

Outside a user interface, processing may be too application-specific to discuss here.

Rationale The use of a single structured element type to represent external resources simplifies processing while allowing a rich set of information to be maintained on each resource.

3.6 Internal References

A single element is provided to support internal references to other elements within an XBEL instance.

The `<alias>` Element

Attributes Only one attribute is needed for `<alias>`, and is required to identify the link referent.

`ref`, ***required***
IDREF value which refers to a `<bookmark>` or `<folder>` element, or the document `<xbel>` element.

Processing Expectations Software which presents bookmarks in a user interface should distinguish aliases from other bookmarks visually, but otherwise allow examination of the referent transparently. Netscape Navigator does this by presenting the bookmark title in an italic font; the appropriate visual distrinction is likely to be dependent on other aspects of the user interface.

Outside of user interface considerations, treatment of aliases is application-specific. However, some guidance may prove useful. When encountering an `<alias>`, an application should only need to traverse the `<alias>` and process the referent if that referent would not otherwise be processed, otherwise, the `<alias>` may usually be ignored. This should become an issue only when the application is processing a portion of the bookmark hierarchy rather than the complete tree.

Rationale Netscape Navigator and Microsoft Internet Explorer bookmarks can include "aliases" to other nodes in the hierarchical structure. Navigator supports only aliases to bookmark nodes, while Internet Explorer also supports aliases to folders.

Navigator's format simply adds the attribute `aliasid` to nodes which are referred to be aliases, and the attribute `aliasof` to the actual alias. All other information is duplicated for each alias of the primary bookmark entry. XBEL uses a distinct element and the ID/IDREF mechanism provided by XML to avoid redundency and support validation.

4 DTD Structure

This section discusses how the DTD itself is organized. This is mostly of interest to the maintainers of XBEL and any descendent document types that may be defined in the future.

4.1 Use of Parameter Entities

Limited use of parameter entities is made in the XBEL DTD. The suffix-notation is adopted from the "XMLspec" DTD report [Mal98]. Specifically, the '`.mix`' suffix is used for entities which define repeatable-or groups of elements, and '`.att`' is used for entities which define attributes.

The `%nodes.mix;` Entity

The `%nodes.mix;` entity lists the element types which may be used to form the nodes of the hierarchical data structure described by an XBEL instance. This entity species a mixture of `<bookmark>`, `<folder>`, `<separator>` and `<alias>` elements.

The `%node.att;` Entity

This entity is used to define attributes for element types which hold the real content of the bookmark data. It is used on the `<bookmark>` and `<folder>` elements. It defines the optional `added` and `id` attributes.

The `%url.att;` Entity

This entity defines the attributes which are available on elements which refer to specific resources. In XBEL 1.0, this is only used on the `<bookmark>` element. It defines a required `href` attribute and the optional attributes `modified` and `visited`.

4.2 Extending the DTD

Extensibility of XBEL relies on three foundations: XML namespaces and the acceptability of well-formed instances, localized parameters entities, and the simplicity of the DTD itself.

The primary expectation for DTD extensions is that new elements and attributes will be introduced and defined using XML namespaces. Though still in the stage of a working draft within the W3C, namespaces offer the most flexible extension mechanism available for XML-based markup languages used in wide-spread deployment. Until validation requirements in the context of namespaces are more clearly defined, XBEL instances using namespaces can apply well-formedness rules as a vehicle for partial validation.

More traditional document type extension uses parameter entities reserved for localization. The XBEL public text provides three such entities as "hooks" to allow local customization. For each of the parameter entities described in Section 4.1, "Use of Parameter Entities," a `%local.`*name*`;` variant is declared and used in the definition of each of the entities described above. This is less flexible than the namespace approach, but allows a new document type to be created which can be used for validation with current tools without having to create a new public text from scratch.

The third foundation for extensibility, the simplicity of the DTD, can be effectively used only by taking a "steal this code" approach to reuse. XBEL is sufficiently simple that it can easily be understood in its entirety, and a variant document type created by crafting a new public text.

4.3 General Entities

The XBEL DTD defines no general entities.

Rationale Since XBEL is intended as an interchange format for software and not as an authoring format, there is no need to support typical entities used to enter special characters. Entities which do not correspond to Unicode characters are too application-specific to predict meaningfully [UC96, UC98].

A Public Text

This section contains the entire public text of the XBEL DTD corresponding to the Formal Public Identifier presented in Section 1.4. No additional external entities are referenced.

```
<!-- This is the XML Bookmarks Exchange Language, version 1.0.  It
     should be used with the formal public identifier:

        +//IDN python.org//DTD XML Bookmark Exchange Language 1.0//EN//XML
     One valid system identifier at which this DTD will remain
     available is:

        http://www.python.org/topics/xml/dtds/xbel-1.0.dtd

     More information the on the DTD, including reference
     documentation, is available at:

        http://www.python.org/topics/xml/xbel/

    Attributes which take date/time values should encode the value
    according to the W3C NOTE on date/time formats:

        http://www.w3.org/TR/NOTE-datetime
  -->


<!-- Customization entities.  Define these before "including" this DTD
     to create "subclassed" DTDs.
  -->
<!ENTITY % local.node.att  "">
<!ENTITY % local.url.att   "">
<!ENTITY % local.nodes.mix "">

<!ENTITY % node.att     "id       ID    #IMPLIED
                         added    CDATA #IMPLIED
                         %local.node.att;">

<!ENTITY % url.att      "href     CDATA #REQUIRED
                         visited  CDATA #IMPLIED
                         modified CDATA #IMPLIED
                         %local.url.att;">

<!ENTITY % nodes.mix    "bookmark|folder|alias|separator
                         %local.nodes.mix;">


<!ELEMENT xbel (title?, info?, desc?, (%nodes.mix;)*)>
<!ATTLIST xbel
            %node.att;
            version  CDATA      #FIXED "1.0"
>
<!ELEMENT title      (#PCDATA)>

<!--=================== Info ======================================-->

<!ELEMENT info (metadata+)>

<!ELEMENT metadata EMPTY>
<!ATTLIST metadata
            owner    CDATA      #REQUIRED
>

<!--=================== Folder ====================================-->

<!ELEMENT folder   (title?, info?, desc?, (%nodes.mix;)*)>
<!ATTLIST folder
            %node.att;
            folded   (yes|no)   'yes'
```

```
>

<!--==================== Bookmark ==================================-->

<!ELEMENT bookmark (title?, info?, desc?)>
<!ATTLIST bookmark
            %node.att;
            %url.att;
>

<!ELEMENT desc       (#PCDATA)>

<!--==================== Separator =================================-->

<!ELEMENT separator EMPTY>

<!--==================== Alias =====================================-->

<!-- <alias> elements correspond to Netscape bookmark aliases.  The
     required "ref" attribute must refer to a <bookmark> or <folder>
     element.  Note that MSIE aliases can refer to folders, so that is
     supported in XBEL.  Applications must be careful about traversing
     aliases to folders to avoid improper recursion through circular
     data structures.
  -->

<!ELEMENT alias EMPTY>
<!ATTLIST alias
            ref       IDREF     #REQUIRED
>
```

References

[BGL98] Dan Brickley, R. V. Guha, and Andrew Layman. Resource description framework (RDF) schema specification. Working draft, World Wide Web Consortium, Aug 1998. http://www.w3.org/TR/WD-rdf-schema.

[BHL98] Tim Bray, Dave Hollander, and Andrew Layman. Namespaces in XML. Working draft, World Wide Web Consortium, Sep 1998. http://www.w3.org/TR/WD-xml-names.

[BPSM98] Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen. Extensible markup language (XML) 1.0. Recommendation, World Wide Web Consortium, Feb 1998. http://www.w3.org/TR/REC-xml.

[Dra] Fred L. Drake, Jr. The XML bookmark exchange language resource page. http://www.python.org/topics/xml/xbel/.

[Dub97] Dublin Core Working Group. Dublin core metadata, Nov 1997. http://purl.oclc.org/metadata/dublin_core/.

[fS88] International Organization for Standardization. *Data elements and interchange formats — Information interchange — Representation of dates and times*. International Organization for Standardization, 1988.

[Hou93] Gary Houston. ISO 8601 date/time representations, Jan 1993. ftp://ftp.informatik.uni-erlangen.de/pub/doc/ISO/ISO8601.ps.Z.

[Kuc] Andrew M. Kuchling. Python and XML processing. http://www.python.org/topics/xml/.

[Kuh98] Markus Kuhn. A summary of the international standard date and time notation, Sep 1998. http://www.cl.cam.ac.uk/ mgk25/iso-time.html.

[LS98] Ora Lassila and Ralph R. Swick. Resource description framework (RDF) model and syntax specification. Working draft, World Wide Web Consortium, Oct 1998. http://www.w3.org/TR/WD-rdf-syntax.

[Mal98] Eve Maler. *W3C XML Specification DTD ('XMLspec')*. World Wide Web Consortium, Sep 1998. http://www.w3.org/XML/1998/06/xmlspec-report-19980910.htm.

[UC96] The Unicode Consotium. *The Unicode Standard, Version 2.0*. Addison-Wesley Developers Press, Reading, MA, 1996.

[UC98] The Unicode Consortium. The Unicode standard, version 2.1. Technical Report 8, The Unicode Consortium, San Jose, CA, Sep 1998. http://www.unicode.org/unicode/reports/tr8.html.

[WW98] Misha Wolf and Charles Wicksteed. Date and time formats. Technical note, World Wide Web Consortium, Sep 1998. http://www.w3.org/TR/WD-rdf-schema.