Tamino Mobile

XML APIs

Implementation of SAX, DOM and XPath

TECHNICAL WHITE PAPER

SOFTWARE AG
THE XML COMPANY

Contents

Software AG
September 2001

Introduction to XML

XML is the universal format for structured documents and data on the Internet. A World Wide Web Consortium (W3C) standard, XML is used to build structures of content and to maintain meta data about that content. What this means in practice is that XML enables organizations to share, exchange and publish data in a versatile and widely-accepted way. For more information on XML, visit www.w3.org/XML/.

XML is widely viewed as an ideal means of building networks of information. XML is now also considered the preferred format for the exchange of data using wireless communication devices. With its flexibility and widespread adoption, XML offers a generalized means of extending applications to include mobile users, both in the consumer and industrial markets. The XML-enabled Tamino Mobile Database Engine (DBE) in handheld device applications will also run smoothly for occasionally connected users.
This XML database engine is referred to as "Tamino Mobile" throughout the remainder of the document.

The abstract definition of XML is called the XML Information Set (Infoset). Its purpose is to provide a consistent set of definitions for referring to the information in a well-formed XML document.

The formal Infoset definition is at www.w3.org/TR/xml-infoset.

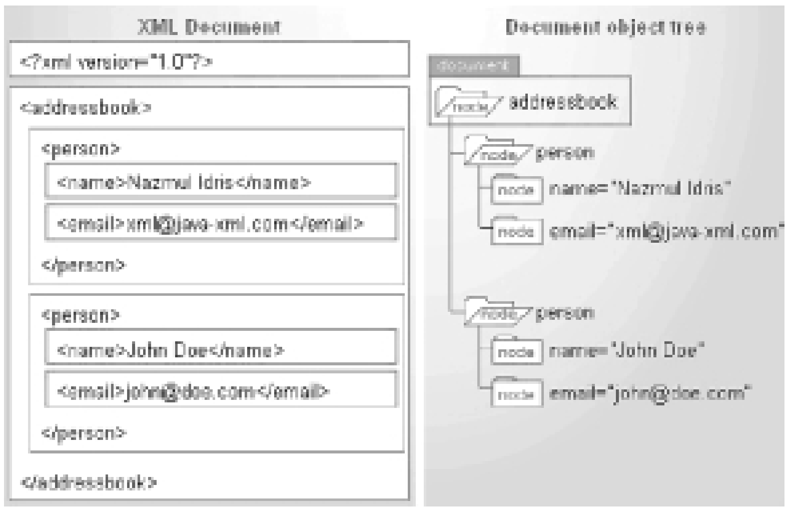
XML documents can have two basic representations:

• Serialized – a stream of characters that conform to defined XML syntax rules. This representation is most suitable for transfer, for browsing and for reading by the human eye.

• Infoset Model – a hierarchy of nodes where each node represents an information element, and where each element can contain other elements or values. This representation is most suitable for navigating in the structure of the document, to search for specific elements or values and to generate subdocuments.

The XML Infoset does not mandate a specific serialization syntax.

The following picture shows serialization beside the hierarchical representation.



```
<?xml version="1.0"?>
<addressbook>
  <person>
    <name>Nazmul Idris</name>
    <email>xml@java-xml.com</email>
  </person>
  <person>
    <name>John Doe</name>
    <email>john@doe.com</email>
  </person>
</addressbook>
```

Fig. 1: Hierarchical structure of a document object

Tamino Mobile

When an XML document is to be represented in a hierarchical structure on a computer, the XML document is either placed in memory or stored in a database. If persistence is needed, then obviously a database to support the structure saves time and gives a more robust solution than having to repeatedly parse the document into memory.

Relational databases are not very well suited for representing XML documents.

They do not support
• order,
• hierarchy,
• irregular structures or
• variable length fields.

Relational databases were designed with tabular data structures as the underlying data model. They are designed for SQL queries, but are not well suited to performing the type of content-related queries in XML-oriented structures.

Tamino Mobile was designed with a focus on
• effective representation,
• navigation and
• searching within XML documents.

The next paragraph indicates key mechanisms found in Tamino Mobile:

Tamino Mobile populated with data can be considered a collection of objects. Each object may be associated with a type (class) which has been declared in the database schema. Objects in Tamino Mobile may be organized in a hierarchical structure. Tamino Mobile is therefore particularly well suited to representing XML documents as a hierarchy of nodes. Tamino Mobile contains mechanisms to establish the structure, to navigate within it and to search across it.

The following features are available in Tamino Mobile:

• A class can have a combination of mandatory attributes and optional attributes.

• Optional attributes can be added dynamically, enabling representation of data structures that are not defined initially through the database schema.

• An object may have zero or more objects as children.

• The children of an object are ordered.

• New children are inserted relative to some other child.

• An object has at most a single-parent object.

• The edge that connects an object to its parent object has a text label that describes the role of the object in its parent.

• Labels can be combined to form paths in the database. Paths can be used to limit the universe in queries on hierarchically structured data.

• Objects with the same parent may have the same label on the edge that connects them to their common parent. This means that a path specifies a set of objects, not a single object. Another way to look at this is to view a label as a subdirectory name, and children attached to the same parent through the same label as part of the same subdirectory. The label represents a subdirectory, and objects can be viewed as files stored in that subdirectory.

• Graphs can be represented through a combination of parent-child structures and use attributes to refer to other objects in the database.

• Indexing provides fast search and navigation.

These features of Tamino Mobile are instrumental to the highly efficient and robust implementation of the XML interfaces. XML documents are stored in Tamino Mobile's XML database as objects in a hierarchical structure. The database structure of an XML document in Tamino Mobile is constructed to resemble the Infoset model as closely as possible. This means that there is a one-to-one correspondence between nodes in the Infoset structure, and the nodes in the tree corresponding to a database document object. For efficiency, attribute nodes in the Infoset are not represented as object instances in the database, so this imposes a structural difference between the Infoset definition and the XML representation in the database. For reasons of efficiency, Infoset attributes are represented as attributes in the object where the attributes appear. To look at the serialized representation of XML documents, these typically contain attributes, values and tags. In Tamino

Mobile's database, tags are stored as objects (tag objects). Attributes and their values are stored as attributes of the tag objects and in the attributes of the tag objects.

Figure 2 illustrates how Tamino Mobile offers SAX, DOM and XPATH to application programmers through

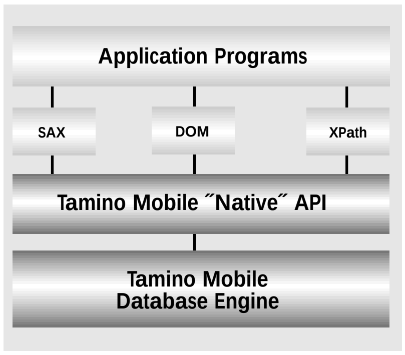


Fig. 2: APIs available for Tamino Mobile

the set of classes and methods defined in these standards. The implementations of SAX, DOM and XPATH are tied to the database engine through native API calls to the database.

SAX

SAX, the Simple API for XML, was designed to allow programmers to access their information without writing a parser.

SAX is a set of abstract programmatic interfaces that projects a document onto a stream of well-known method calls. Tamino Mobile uses the open source Expat-parser to parse the stream and read the integrated tags, attributes, values, text, and structure of XML documents. Tamino Mobile's SAX module contains methods to transform the document into a hierarchy of objects and attributes that will represent the document.

SAX describes an event-driven interface to the process of parsing XML documents. SAX is an API in the public domain, developed by individuals on the XML-DEV mailing list. It does not have a formal specification document, but is defined by a public domain implementation using the Java™ programming language. An XML parser is SAX-conformant if it implements the interface defined by this public domain implementation.

An event-driven interface provides a mechanism for notifications to the application code as the underlying parser recognizes XML syntactic constructions in the document.

USING SAX TO IMPORT XML DOCUMENTS AND DATA

SAX allows you to access the information in your XML document, not as a tree of nodes, but as a sequence of events. The Tamino Mobile SAX implementation listens to SAX events, which are generated by the SAX parser as it is reading the XML document. It utilizes these events to create objects in the database.

SAX will fire an event for every open tag and every close tag. It also fires events for #PCDATA and CDATA sections, processing instructions, DTDs, comments, and so on. Figure 3 illustrates how the Tamino Mobile SAX implementation is used to import an XML document into a Tamino Mobile database.

USING SAX TO RETRIEVE XML DOCUMENTS AND DATA FROM TAMINO MOBILE

An XML document stored in a Tamino Mobile database can be retrieved in part or whole, using our SAX implementation. You can then write application programs using the Tamino Mobile SAX interface as an event-driven feeder of the elements of the document. Note that the elements are fed sequentially. Navigating back and forth is not possible using SAX. The application program is of course free to choose how to respond to the events. It may, for example, choose to serialize the document, i.e. rebuild the document; it may choose to look for a specific tag or it may perform statistics. Figure 4 shows how Tamino Mobile's SAX implementation interfaces with your program and the Tamino Mobile database engine.

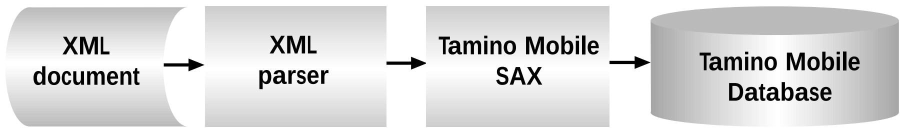


Fig. 3: Tamino Mobile SAX implementation: Import of XML documents

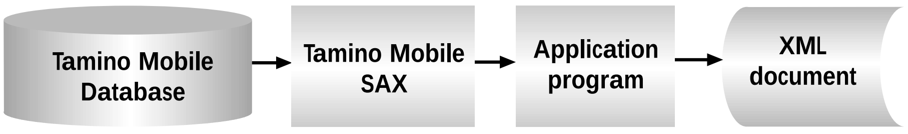


Fig. 4: Tamino Mobile SAX implementation: Retrieval of XML documents

DOM

Just as an XML parser in general and SAX in particular add a layer of abstraction over the actual textual representation of the XML document, the Document Object Model (DOM) adds a layer of abstraction on top of the entire document. DOM standardizes the object model representing an XML document and defines a language- and platform-neutral interface to the structure and style of XML documents, which a process may dynamically access and update. Elements are considered as nodes in a tree instead of being composed by start and end tags. Nodes may have parents and children, and they may have internal properties that can be modified using objects and methods.

DOM gives programs access to the information stored in an XML document as a hierarchical object model. DOM regards the document as a tree of nodes (based on the structure and information in the XML document). Programmers can access the information by interacting with this tree of nodes. DOM defines a set of abstract interfaces that models a document conforming to the XML Infoset specifications. Tamino Mobile’s DOM implementation consists of a set of methods to provide access to the document stored as a hierarchy of objects in the database. Our DOM implementation supports applications written in C++.

The Document Object Model specifies a tree-based representation for an XML document. A top-level document instance is the root of the tree and has a single child that is the top-level element instance; this element has children nodes representing the content and any sub-elements. These sub-elements may have further children, many generations deep. Functions are defined which let you traverse the resulting tree any way you like, access element and attribute values, insert and delete nodes, and convert the tree back into XML.

The DOM is useful for modifying XML documents because you can create a DOM tree, modify it by adding new nodes and moving subtrees around and then produce a new XML document as output. You can also construct a DOM tree yourself, and convert it to XML; this is often a more flexible way of producing XML output than simply writing <tag1>...</tag1> to a file.

For some classes of applications, using SAX or interfacing directly with an XML parser may be the ideal way to process XML documents. If the application is expected to handle XML documents – with as little latency as possible, or to handle documents too large to fit in memory – processing each event as it occurs in the document is needed. The problem with using SAX is that the application has to set up event handlers for all elements the application cares about and build its own data structures on-the-fly as the events occur. Rather than responding to each event, it would be easier if the entire tree was already loaded into memory and it was possible to navigate the tree and manipulate parts of it in a simple way.

The Tamino Mobile storage model is designed to be as close to the DOM as possible, but still simple enough to be efficient when outputting data sequentially, i.e. through SAX. Since Tamino Mobile stores XML data in structures that are close to the DOM, the DOM interface (or SAX handler) does not have to load an entire XML document into memory before the user may access it. The DOM methods and objects can be accessed while the document is in the buffer cache. This is a major difference from the approach necessary when the XML document is stored as a sequential chain of entities.

Tamino Mobile's XML Interface is influenced significantly by Apache's DOM interface, Xerces. This allows users who are familiar with Xerces to get started quickly with Tamino Mobile's XML Interface. It also means that that projects using Xerces as the underlying interface can be easily ported to the Tamino Mobile XML Interface. However, the underlying architecture is completely different, since the Tamino Mobile DOM interface is based on a persistent database medium and Xerces is based on a dynamic memory representation.

XPath

XPath, the XML Path Language, is a language for selecting a set of nodes in an XML document. The syntax of the language is path-based. Tamino Mobile's XPath module contains logic to interpret the query, transform the query into a series of calls to Tamino Mobile's API methods, and present the resulting set of nodes.
The primary purpose of XPath is to address parts of an XML document. In support of this primary purpose, it also provides basic facilities for the manipulation of strings, numbers and Boolean values. XPath uses a compact, non-XML syntax to facilitate its use within URLs and XML attribute values. XPath operates on the abstract, logical structure of an XML document, rather than its surface syntax. It got its name by using a path notation such as a URL for navigating through the hierarchical structure of an XML document.

The primary syntactic construct in XPath is the expression.

An expression is evaluated to yield an object of one of the following four basic types:

- node-set (an unordered collection of nodes without duplicates)
- Boolean (true or false)
- number (a floating-point number)
- string (a sequence of UCS characters)

XPath is used to extract parts of an XML document. In Tamino Mobile the XML document is represented as a hierarchy of objects (nodes). When an XPath statement is used in a program, the expression and the context node are given as parameters. The XPath parser evaluates the expression and transforms the expression into a query to Tamino Mobile. The result of the query is an unordered set of nodes (which can be empty) that satisfies the expression's criteria.

The query is implemented using the LookUp functions of Tamino Mobile. The indexing scheme of the database engine provides high performance searching. The result set is provided through the cursor-oriented collections in Tamino Mobile.

More information on
Tamino XML Server:
http://www.softwareag.com/tamino

Tamino Community:
http://www.softwareag.com/developer

Download XML Starter Kit:
http://www.xmlstarterkit.com

Software AG
Corporate Headquarters
Uhlandstraße 12
64297 Darmstadt/Germany
Tel: +49-61 51-92-0
Fax: +49-61 51-92-11 91
www.softwareag.com

TMO/WP02E0901 229

SOFTWARE AG
THE XML COMPANY