Applying the DOM Object model to structured binary file formats

The Document Object Model (DOM) is a general API for accessing SGML and XML documents that was developped by the W3 Consortium.

The DOM API works on a tree composed of nodes, where the entire file is parsed and a hierarchy of the elements are created as a tree representation.

There is no reason why a subset of this API could not be made available for binary structured file formats. The core API, using the Node interface is ideal for representing such structured file formats as RIFF, IFF, OLE, TIFF and JPEG files.

This document describes the changes that have to be made to the Core and XPath DOM specifications so they can be used with binary structured file formats. It assumes that no specific XML/HTML interface is used.

DOM Core specification changes

DOM Basic types

DOM Basic types	***Availability***
DOMString	x
DOMTimeStamp	x
DOMUserData	x
DOMObject	x

DOM Interfaces

Only a subset of the fundamental interfaces are available in this recommendation. None of the extended interfaces are available (it would make no sense to have an XML/SGML interface for binary files).

DOM Core interface	***Availability***
DOMException	x
DOMStringList	N/A
NameList	N/A
DOMImplementationList	N/A
DOMImplementationSource	N/A
DOMImplementation	N/A
DocumentFragment	N/A
Document	Partial

DOM Core interface	*Availability*
Node	x
NodeList	x
NamedNodeMap	???
CharacterData	N/A
Attr	N/A
Element	N/A
Text	N/A
Comment	N/A
TypeInfo	N/A
UserDataHandler	N/A
DOMError	N/A
DOMErrorHandler	N/A
DOMLocator	N/A
DOMConfiguration	N/A

DOM XPath Changes

The only types of nodes that are supported in XPath are the Element nodes and the Text nodes. The other nodes are not supported for evident reasons.

DOM XPath interface	*Availability*
XPathException	x
XPathEvaluator	Partial
XPathExpression	N/A
XPathNSResolver	N/A
XPathResult	Partial
XPathNamespace	N/A

Because the file types are binary, the Node.nodeValue type must be changed to represent binary data. Therefore in this model, instead of being considered an array of unicode characters (of type DOMString), it is considered an array of bytes.

The Node.nodeName is still represented as a DOMString.

Node type support

Node type	*Comment*
ATTRIBUTE_NODE	This can never occur in a structured binary file format.
CDATA_SECTION_NODE	This can never occur in a structured binary file format.
COMMENT_NODE	This can never occur in a structured binary file format.
DOCUMENT_FRAGMENT_NODE	Currently unsupported.
DOCUMENT_NODE	Always present, actually is the root node.
DOCUMENT_TYPE_NODE	Always present, actually gives the type of the file format, it is not related at all to the DTD or DOCTYPE.
ELEMENT_NODE	Usually present, see below for more information on how this value is interpreted.
ENTITY_NODE	This can never occur in a structured binary file format.
ENTITY_REFERENCE_NODE	This can never occur in a structured binary file format.
NOTATION_NODE	This can never occur in a structured binary file format, since DTD's cannot be included in the actual file.
PROCESSING_INSTRUCTION_NODE	This can never occur in a structured binary file format.
TEXT_NODE	In this recommendation, this node represents the actual binary data of a leaf node. As stated earlier, the Node.nodeValue attribute is a an array of bytes instead of being an array of unicode characters.

ELEMENT_NODE Explanations

Most structured binary file types are either based on a directory approach, or on chunk based approach.

directory based file types

This type of file is based on one or more directory tables, where each directory tables contains one or more directory entries.

A directory entry specifies where the actual data for a type of information is located, This is the scheme used in both TIFF files and OLE (Microsoft Word) files.

TIFF files

In TIFF files, each directory entry has a numeric id value which givesinformation on the type of entry. Each of these id values can be considered an ELEMENT_NODE with nodeName be equal to either equal to the mnemonic name (as specified in the TIFF specification) or directly as the identifier number (converted to a DOMString). Node.nodeValue would always be equal to null.The leaf node of the ELEMENT_NODE would always be a TEXT_NODE that contains the data associated with this directory entry.

OLE files

OLE documents are represented directly as a file system where each storage element (directory) is represented as an ELEMENT_NODE that can contain other ELEMENT_NODE nodes if it contains other directories or files. It is a leaf node, if there are no children in the directory. The nodeName is the actual name of the directory.

Files are also ELEMENT_NODE nodes, where their child is the data of the file, represented as a TEXT_NODE. Node.noddeName contains the name of the file.

Chunk based file types

This kind of structured file is composed of data chunks, each of them being precedeed by a header giving the chunk type and size, followed by the actual data, this is the approach used in JPEG, RIFF, IFF and PNG file.

In the case of IFF, RIFF and PNG files, the Node.nodeName will be equal to the four character signature chunk name. Usually (when this is not a chunk that indicates nesting), the child node of this element node shall be a TEXT_NODE with the actual data for this chunk For JPEG files, the Node.nodeName value shall be equal to the 3 character mnemonic for the markers.