

IXML v1.2

Contents

1	Introduction	4
2	License	5
3	BOOL	6
4	DOM Interfaces	7
4.1	Interface <i>Node</i>	7
4.2	Interface <i>Attr</i>	18
4.3	Interface <i>CDATASection</i>	19
4.4	Interface <i>Document</i>	20
4.5	Interface <i>Element</i>	32
4.6	Interface <i>NamedNodeMap</i>	43
4.7	Interface <i>NodeList</i>	47
5	IXML API	49

Linux DOM2 XML Parser Version 1.2

Copyright (C) 2000-2003 Intel Corporation ALL RIGHTS RESERVED

Revision 1.2.1 (Sun Aug 14 20:27:19 2005)

Introduction

The Linux DOM2 XML Parser Version 1.2 (IXML) is a lightweight, portable XML parser supporting the standard Document Object Model (DOM) Level 2 interfaces. The parser uses a C-style interface, making it ideal for small, embedded applications. This document describes the interfaces supported by IXML 1.2, referencing the W3C DOM2 recommendations when necessary, and the additional utility application programming interfaces (APIs) that it supports.

Note that this document assumes that the reader has a copy of the DOM2-Core recommendation. Refer to the link below to obtain a copy. Only a brief description is included here and the reader is pointed to the DOM2-Core recommendation for more details. This document does, however, clarify IXML-specific behavior when the recommendation is unclear.

About DOM

The Document Object Model (DOM) is a set of interfaces that give a programmatic interface to documents. It provides a platform-neutral and language-neutral interface for random access and updating elements inside XML documents. DOM Level 1 provided the basic interfaces to access document elements. DOM Level 2 extended the interfaces to provide proper support for XML namespaces.

The latest DOM 2 recommendation is maintained by W3C and is available from <http://www.w3.org/TR/DOM-Level-2-Core>.

License

Copyright (c) 2000-2003 Intel Corporation All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither name of Intel Corporation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTEL OR CONTRIBUTORS 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 USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

3

```
typedef int BOOL
```

DOM Interfaces

Names

4.1	Interface <i>Node</i>	7
4.2	Interface <i>Attr</i>	18
4.3	Interface <i>CDATASection</i>	19
4.4	Interface <i>Document</i>	20
4.5	Interface <i>Element</i>	32
4.6	Interface <i>NamedNodeMap</i>	43
4.7	Interface <i>NodeList</i>	47

The Document Object Model consists of a set of objects and interfaces for accessing and manipulating documents. IXML does not implement all the interfaces documented in the DOM2-Core recommendation but defines a subset of the most useful interfaces. A description of the supported interfaces and methods is presented in this section.

For a complete discussion on the object model, the object hierarchy, etc., refer to section 1.1 of the DOM2-Core recommendation.

Interface *Node*

Names

4.1.1	const DOMString	ixmlNode_getNodeName (IXML_Node* nodeptr) <i>Returns the name of the Node, depending on what type of Node it is, in a read-only string.</i>	9
4.1.2	DOMString	ixmlNode_getNodeValue (IXML_Node* nodeptr) <i>Returns the value of the Node as a string.</i>	10
4.1.3	int	ixmlNode_setNodeValue (IXML_Node* nodeptr, char* newNodeValue) <i>Assigns a new value to a Node.</i>	10
4.1.4	const unsigned short	ixmlNode_getNodeType (IXML_Node* nodeptr) <i>Retrieves the type of a Node.</i>	10
4.1.5	IXML_Node*	ixmlNode_getParentNode (IXML_Node* nodeptr) <i>Retrieves the parent Node for a Node.</i>	11
4.1.6	IXML_NodeList*		

		ixmlNode_getChildNodes (IXML_Node* nodeptr)	<i>Retrieves the list of children of a Node in a NodeList structure.</i>	11
4.1.7	IXML_Node*	ixmlNode_getFirstChild (IXML_Node* nodeptr)	<i>Retrieves the first child Node of a Node.</i>	12
4.1.8	IXML_Node*	ixmlNode_getLastChild (IXML_Node* nodeptr)	<i>Retrieves the last child Node of a Node.</i>	12
4.1.9	IXML_Node*	ixmlNode_getPreviousSibling (IXML_Node* nodeptr)	<i>Retrieves the sibling Node immediately preceding this Node.</i>	12
4.1.10	IXML_Node*	ixmlNode_getNextSibling (IXML_Node* nodeptr)	<i>Retrieves the sibling Node immediately following this Node.</i>	13
4.1.11	IXML_NamedNodeMap*	ixmlNode_getAttributes (IXML_Node* nodeptr)	<i>Retrieves the attributes of a Node, if it is an Element node, in a NamedNodeMap structure.</i>	13
4.1.12	IXML_Document*	ixmlNode_getOwnerDocument (IXML_Node* nodeptr)	<i>Retrieves the document object associated with this Node.</i>	13
4.1.13	const DOMString	ixmlNode_getNamespaceURI (IXML_Node* nodeptr)	<i>Retrieves the namespace URI for a Node as a DOMString.</i>	14
4.1.14	DOMString	ixmlNode_getPrefix (IXML_Node* nodeptr)	<i>Retrieves the namespace prefix, if present.</i>	14
4.1.15	const DOMString	ixmlNode_getLocalName (IXML_Node* nodeptr)	<i>Retrieves the local name of a Node, if present.</i>	14
4.1.16	int	ixmlNode_insertBefore (IXML_Node* nodeptr, IXML_Node* newChild, IXML_Node* refChild)	<i>Inserts a new child Node before the existing child Node.</i>	15
4.1.17	int	ixmlNode_replaceChild (IXML_Node* nodeptr, IXML_Node* newChild, IXML_Node* oldChild, IXML_Node** returnNode)		

			<i>Replaces an existing child Node with a new child Node in the list of children of a Node.</i>	16
4.1.18	int	ixmlNode_removeChild	(IXML_Node* nodeptr, IXML_Node* oldChild, IXML_Node** returnNode) <i>Removes a child from the list of children of a Node.</i>	16
4.1.19	int	ixmlNode_appendChild	(IXML_Node* nodeptr, IXML_Node* newChild) <i>Appends a child Node to the list of children of a Node.</i>	16
4.1.20	BOOL	ixmlNode_hasChildNodes	(IXML_Node* nodeptr) <i>Queries whether or not a Node has children.</i>	17
4.1.21	IXML_Node*	ixmlNode_cloneNode	(IXML_Node* nodeptr, BOOL deep) <i>Clones a Node.</i>	17
4.1.22	BOOL	ixmlNode_hasAttributes	(IXML_Node* node) <i>Queries whether this Node has attributes.</i>	18
4.1.23	void	ixmlNode_free	(IXML_Node* IXML_Node) <i>Frees a Node and all Nodes in its subtree.</i>	18

The **Node** interface forms the primary datatype for all other DOM objects. Every other interface is derived from this interface, inheriting its functionality. For more information, refer to DOM2-Core page 34.

4.1.1

```
const DOMString ixmlNode_getNodeName (IXML_Node* nodeptr )
```

*Returns the name of the **Node**, depending on what type of **Node** it is, in a read-only string.*

Returns the name of the **Node**, depending on what type of **Node** it is, in a read-only string. Refer to the table in the DOM2-Core for a description of the node names for various interfaces.

Return Value: [const DOMString] A constant **DOMString** of the node name.

Parameters: nodeptr Pointer to the node to retrieve the name.

4.1.2

```
DOMString ixmlNode_getNodeValue (IXML_Node* nodeptr )
```

*Returns the value of the **Node** as a string.*

Returns the value of the **Node** as a string. Note that this string is not a copy and modifying it will modify the value of the **Node**.

Return Value: [DOMString] A **DOMString** of the **Node** value.
Parameters: nodeptr Pointer to the **Node** to retrieve the value.

4.1.3

```
int ixmlNode_setNodeValue (IXML_Node* nodeptr, char* newNodeValue )
```

*Assigns a new value to a **Node**.*

Assigns a new value to a **Node**. The **newNodeValue** string is duplicated and stored in the **Node** so that the original does not have to persist past this call.

Return Value: [int] An integer representing one of the following:

- IXML_SUCCESS: The operation completed successfully.
- IXML_INVALID_PARAMETER: The **Node*** is not a valid pointer.
- IXML_INSUFFICIENT_MEMORY: Not enough free memory exists to complete this operation.

Parameters: nodeptr The **Node** to which to assign a new value.
 newNodeValue The new value of the **Node**.

4.1.4

```
const unsigned short ixmlNode_getNodeType (IXML_Node* nodeptr )
```

*Retrieves the type of a **Node**.*

Retrieves the type of a **Node**. The defined **Node** constants are:

- eATTRIBUTE_NODE
- eCDATA_SECTION_NODE
- eCOMMENT_NODE

- eDOCUMENT_FRAGMENT_NODE
- eDOCUMENT_NODE
- eDOCUMENT_TYPE_NODE
- eELEMENT_NODE
- eENTITY_NODE
- eENTITY_REFERENCE_NODE
- eNOTATION_NODE
- ePROCESSING_INSTRUCTION_NODE
- eTEXT_NODE

Return Value: [const unsigned short] An integer representing the type of the **Node**.

Parameters: nodeptr The **Node** from which to retrieve the type.

4.1.5

IXML_Node* **ixmlNode_getParentNode** (IXML_Node* nodeptr)

*Retrieves the parent **Node** for a **Node**.*

Retrieves the parent **Node** for a **Node**.

Return Value: [Node*] A pointer to the parent **Node** or NULL if the **Node** has no parent.

Parameters: nodeptr The **Node** from which to retrieve the parent.

4.1.6

IXML_NodeList* **ixmlNode_getChildNodes** (IXML_Node* nodeptr)

*Retrieves the list of children of a **Node** in a **NodeList** structure.*

Retrieves the list of children of a **Node** in a **NodeList** structure. If a **Node** has no children, **ixmlNode_getChildNodes** returns a **NodeList** structure that contains no **Nodes**.

Return Value: [NodeList*] A **NodeList** of the children of the **Node**.

Parameters: nodeptr The **Node** from which to retrieve the children.

4.1.7

IXML_Node* **ixmlNode_getFirstChild** (IXML_Node* nodeptr)

*Retrieves the first child **Node** of a **Node**.*

Retrieves the first child **Node** of a **Node**.

Return Value:	[Node*]	A pointer to the first child Node or NULL if the Node does not have any children.
Parameters:	nodeptr	The Node from which to retrieve the first child.

4.1.8

IXML_Node* **ixmlNode_getLastChild** (IXML_Node* nodeptr)

*Retrieves the last child **Node** of a **Node**.*

Retrieves the last child **Node** of a **Node**.

Return Value:	[Node*]	A pointer to the last child Node or NULL if the Node does not have any children.
Parameters:	nodeptr	The Node from which to retrieve the last child.

4.1.9

IXML_Node* **ixmlNode_getPreviousSibling** (IXML_Node* nodeptr)

*Retrieves the sibling **Node** immediately preceding this **Node**.*

Retrieves the sibling **Node** immediately preceding this **Node**.

Return Value:	[Node*]	A pointer to the previous sibling Node or NULL if no such Node exists.
Parameters:	nodeptr	The Node for which to retrieve the previous sibling.

4.1.10

```
IXML_Node* ixmlNode_getNextSibling (IXML_Node* nodeptr )
```

*Retrieves the sibling **Node** immediately following this **Node**.*

Retrieves the sibling **Node** immediately following this **Node**.

Return Value: [Node*] A pointer to the next sibling **Node** or NULL if no such **Node** exists.

Parameters: nodeptr The **Node** from which to retrieve the next sibling.

4.1.11

```
IXML_NamedNodeMap* ixmlNode_getAttributes (IXML_Node* nodeptr )
```

*Retrieves the attributes of a **Node**, if it is an **Element** node, in a **NamedNodeMap** structure.*

Retrieves the attributes of a **Node**, if it is an **Element** node, in a **NamedNodeMap** structure.

Return Value: [NamedNodeMap*] A **NamedNodeMap** of the attributes or NULL.

Parameters: nodeptr The **Node** from which to retrieve the attributes.

4.1.12

```
IXML_Document* ixmlNode_getOwnerDocument (IXML_Node* nodeptr )
```

*Retrieves the document object associated with this **Node**.*

Retrieves the document object associated with this **Node**. This owner document **Node** allows other **Nodes** to be created in the context of this document. Note that **Document** nodes do not have an owner document.

Return Value: [Document*] A pointer to the owning **Document** or NULL, if the **Node** does not have an owner.

Parameters: nodeptr The **Node** from which to retrieve the owner document.

4.1.13

```
const DOMString xmlNode_getNamespaceURI (IXML_Node* nodeptr
)
```

*Retrieves the namespace URI for a **Node** as a **DOMString**.*

Retrieves the namespace URI for a **Node** as a **DOMString**. Only **Nodes** of type **eELEMENT_NODE** or **eATTRIBUTE_NODE** can have a namespace URI. **Nodes** created through the **Document** interface will only contain a namespace if created using **xmlDocument_createElementNS**.

Return Value: [const DOMString] A **DOMString** representing the URI of the namespace or NULL.

Parameters: nodeptr The **Node** for which to retrieve the namespace.

4.1.14

```
DOMString xmlNode_getPrefix (IXML_Node* nodeptr )
```

Retrieves the namespace prefix, if present.

Retrieves the namespace prefix, if present. The prefix is the name used as an alias for the namespace URI for this element. Only **Nodes** of type **eELEMENT_NODE** or **eATTRIBUTE_NODE** can have a prefix. **Nodes** created through the **Document** interface will only contain a prefix if created using **xmlDocument_createElementNS**.

Return Value: [DOMString] A **DOMString** representing the namespace prefix or NULL.

Parameters: nodeptr The **Node** from which to retrieve the prefix.

4.1.15

```
const DOMString xmlNode_getLocalName (IXML_Node* nodeptr )
```

*Retrieves the local name of a **Node**, if present.*

Retrieves the local name of a **Node**, if present. The local name is the tag name without the namespace prefix. Only **Nodes** of type **eELEMENT_NODE** or **eATTRIBUTE_NODE** can have a local name. **Nodes** created through the **Document** interface will only contain a local name if created using **xmlDocument_createElementNS**.

Return Value: [const DOMString] A **DOMString** representing the local name of the **Element** or NULL.

Parameters: nodeptr The **Node** from which to retrieve the local name.

4.1.16

```
int ixmlNode_insertBefore (IXML_Node*   nodeptr,      IXML_Node*
                           newChild,  IXML_Node* refChild )
```

*Inserts a new child **Node** before the existing child **Node**.*

Inserts a new child **Node** before the existing child **Node**. **refChild** can be NULL, which inserts **newChild** at the end of the list of children. Note that the **Node** (or **Nodes**) in **newChild** must already be owned by the owner document (or have no owner at all) of **nodeptr** for insertion. If not, the **Node** (or **Nodes**) must be imported into the document using **ixmlDocument_importNode**. If **newChild** is already in the tree, it is removed first.

Return Value: [int] An integer representing one of the following:

- IXML_SUCCESS: The operation completed successfully.
- IXML_INVALID_PARAMETER: Either **nodeptr** or **newChild** is NULL.
- IXML_HIERARCHY_REQUEST_ERR: The type of the **Node** does not allow children of the type of **newChild**.
- IXML_WRONG_DOCUMENT_ERR: **newChild** has an owner document that does not match the owner of **nodeptr**.
- IXML_NO_MODIFICATION_ALLOWED_ERR: **nodeptr** is read-only or the parent of the **Node** being inserted is read-only.
- IXML_NOT_FOUND_ERR: **refChild** is not a child of **nodeptr**.

Parameters:

nodeptr	The parent of the Node before which to insert the new child.
newChild	The Node to insert into the tree.
refChild	The reference child where the new Node should be inserted. The new Node will appear directly before the reference child.

4.1.17

```
int ixmlNode_replaceChild (IXML_Node*   nodeptr,      IXML_Node*
                           newChild,      IXML_Node* oldChild,
                           IXML_Node** returnNode )
```

*Replaces an existing child **Node** with a new child **Node** in the list of children of a **Node**.*

Parameters:	nodeptr	The parent of the Node which contains the child to replace.
	newChild	The child with which to replace oldChild .
	oldChild	The child to replace with newChild .
	returnNode	Pointer to a Node to place the removed old-Child Node .

4.1.18

```
int ixmlNode_removeChild (IXML_Node* nodeptr,  IXML_Node* old-
                          Child, IXML_Node** returnNode )
```

*Removes a child from the list of children of a **Node**.*

Removes a child from the list of children of a **Node**. **returnNode** will contain the **oldChild Node**, appropriately removed from the tree (i.e. it will no longer have an owner document).

Return Value: [int] An integer representing one of the following:

- **IXML_SUCCESS**: The operation completed successfully.
- **IXML_INVALID_PARAMETER**: Either **nodeptr** or **oldChild** is NULL.
- **IXML_NO_MODIFICATION_ALLOWED_ERR**: **nodeptr** or its parent is read-only.
- **IXML_NOT_FOUND_ERR**: **oldChild** is not among the children of **nodeptr**.

Parameters:	nodeptr	The parent of the child to remove.
	oldChild	The child Node to remove.
	returnNode	Pointer to a Node to place the removed old-Child Node .

4.1.19

```
int ixmlNode_appendChild (IXML_Node*  nodeptr,  IXML_Node*
                          newChild )
```

*Appends a child **Node** to the list of children of a **Node**.*

Appends a child **Node** to the list of children of a **Node**. If **newChild** is already in the tree, it is removed first.

Return Value: [int] An integer representing one of the following:

- **IXML_SUCCESS:** The operation completed successfully.
- **IXML_INVALID_PARAMETER:** Either **nodeptr** or **newChild** is NULL.
- **IXML_HIERARCHY_REQUEST_ERR:** **newChild** is of a type that cannot be added as a child of **nodeptr** or **newChild** is an ancestor of **nodeptr**.
- **IXML_WRONG_DOCUMENT_ERR:** **newChild** was created from a different document than **nodeptr**.
- **IXML_NO_MODIFICATION_ALLOWED_ERR:** **nodeptr** is a read-only **Node**.

Parameters:

nodeptr	The Node in which to append the new child.
newChild	The new child to append.

4.1.20

```

BOOL ixmlNode_hasChildNodes (IXML_Node* nodeptr )

```

*Queries whether or not a **Node** has children.*

Queries whether or not a **Node** has children.

Return Value: [BOOL] TRUE if the **Node** has one or more children otherwise FALSE.

Parameters: **nodeptr** The **Node** to query for children.

4.1.21

```

IXML_Node* ixmlNode_cloneNode (IXML_Node* nodeptr, BOOL deep
                                )

```

*Clones a **Node**.*

Clones a **Node**. The new **Node** does not have a parent. The **deep** parameter controls whether the subtree of the **Node** is also cloned. For details on cloning specific types of **Nodes**, refer to the DOM2-Core recommendation.

Return Value: [Node*] A clone of **nodeptr** or NULL.
Parameters: **nodeptr** The **Node** to clone.
deep TRUE to clone the subtree also or FALSE to clone only **nodeptr**.

4.1.22

BOOL **ixmlNode_hasAttributes** (IXML_Node* node)

*Queries whether this **Node** has attributes.*

Queries whether this **Node** has attributes. Note that only **Element** nodes have attributes.

Return Value: [BOOL] TRUE if the **Node** has attributes otherwise FALSE.
Parameters: **node** The **Node** to query for attributes.

4.1.23

void **ixmlNode_free** (IXML_Node* IXML_Node)

*Frees a **Node** and all **Nodes** in its subtree.*

Frees a **Node** and all **Nodes** in its subtree.

Return Value: [void] This function does not return a value.
Parameters: IXML_Node The **Node** to free.

4.2

Interface *Attr*

Names

4.2.1 void **ixmlAttr_free** (IXML_Attr* attrNode)
*Frees an **Attr** node. 19*

The **Attr** interface represents an attribute of an **Element**. The document type definition (DTD) or schema usually dictate the allowable attributes and values for a particular element. For more information, refer to the *Interface Attr* section in the DOM2-Core.

4.2.1

```
void ixmlAttr_free (IXML_Attr* attrNode )
```

*Frees an **Attr** node.*

Frees an **Attr** node.

Return Value: [void] This function does not return a value.

Parameters: attrNode The **Attr** node to free.

4.3

Interface *CDATASection*

Names

- | | | | | |
|-------|------|------------------------------------------------------------|----------------------------------------------------|----|
| 4.3.1 | void | ixmlCDATASection_init (IXML_CDATASection* nodeptr) | <i>Initializes a CDATASection node. ...</i> | 19 |
| 4.3.2 | void | ixmlCDATASection_free (IXML_CDATASection* nodeptr) | <i>Frees a CDATASection node.</i> | 20 |

The **CDATASection** is used to escape blocks of text containing characters that would otherwise be regarded as markup. CDATA sections cannot be nested. Their primary purpose is for including material such XML fragments, without needing to escape all the delimiters. For more information, refer to the *Interface CDATASection* section in the DOM2-Core.

4.3.1

```
void ixmlCDATASection_init (IXML_CDATASection* nodeptr )
```

*Initializes a **CDATASection** node.*

Initializes a **CDATASection** node.

Return Value: [void] This function does not return a value.

Parameters: nodeptr The **CDATASection** node to initialize.

4.3.2

```
void ixmlCDATASection_free (IXML_CDATASection* nodeptr )
```

*Frees a **CDATASection** node.*

Frees a **CDATASection** node.

Return Value: [void] This function does not return a value.

Parameters: nodeptr The **CDATASection** node to free.

4.4

Interface *Document*

Names

- | | | | | |
|-------|----------------|-----------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------|----|
| 4.4.1 | void | ixmlDocument_init (IXML_Document* nodeptr) | <i>Initializes a Document node.</i> | 23 |
| 4.4.2 | int | ixmlDocument_createDocumentEx (IXML_Document** doc
) | <i>Creates a new empty Document node.</i> | 23 |
| 4.4.3 | IXML_Document* | ixmlDocument_createDocument () | <i>Creates a new empty Document node.</i> | 24 |
| 4.4.4 | int | ixmlDocument_createElementEx (IXML_Document* doc,
DOMString tagName,
IXML_Element**
rtElement) | <i>Creates a new Element node with the
given tag name.</i> | 24 |
| 4.4.5 | IXML_Element* | ixmlDocument_createElement (IXML_Document* doc,
DOMString tagName) | <i>Creates a new Element node with the
given tag name.</i> | 25 |
| 4.4.6 | int | ixmlDocument_createTextNodeEx (IXML_Document* doc,
DOMString data,
IXML_Node** textNode
) | <i>Creates a new Text node with the given
data.</i> | 25 |
| 4.4.7 | IXML_Node* | ixmlDocument_createTextNode (IXML_Document* doc,
DOMString data) | | |

			<i>Creates a new Text node with the given data.</i>	26
4.4.8	int	ixmlDocument_createCDATASectionEx (IXML_Document* doc, DOMString data, IXML_CDATASection** cdNode)	<i>Creates a new CDATASection node with given data.</i>	26
4.4.9	IXML_CDATASection*	ixmlDocument_createCDATASection (IXML_Document* doc, DOMString data)	<i>Creates a new CDATASection node with given data.</i>	27
4.4.10	IXML_Attr*	ixmlDocument_createAttribute (IXML_Document* doc, char* name)	<i>Creates a new Attr node with the given name.</i>	27
4.4.11	int	ixmlDocument_createAttributeEx (IXML_Document* doc, char* name, IXML_Attr** attrNode)	<i>Creates a new Attr node with the given name.</i>	27
4.4.12	IXML_NodeList*	ixmlDocument_getElementsByTagName (IXML_Document* doc, DOMString tagName)	<i>Returns a NodeList of all Elements that match the given tag name in the order in which they were encountered in a preorder traversal of the Document tree.</i>	28
4.4.13	int	ixmlDocument_createElementNSEx (IXML_Document* doc, DOMString namespaceURI, DOMString qualifiedName, IXML_Element** rtElement)	<i>Creates a new Element node in the given qualified name and namespace URI.</i>	28
4.4.14	IXML_Element*			

		ixmlDocument_createElementNS (IXML_Document* doc, DOMString namespaceURI, DOMString qualifiedName) <i>Creates a new Element node in the given qualified name and namespace URI.</i>	29
4.4.15	int	ixmlDocument_createAttributeNSEx (IXML_Document* doc, DOMString namespaceURI, DOMString qualifiedName, IXML_Attr** attrNode) <i>Creates a new Attr node with the given qualified name and namespace URI.</i>	29
4.4.16	IXML_Attr*	ixmlDocument_createAttributeNS (IXML_Document* doc, DOMString namespaceURI, DOMString qualifiedName) <i>Creates a new Attr node with the given qualified name and namespace URI.</i>	30
4.4.17	IXML_NodeList*	ixmlDocument_getElementsByTagNameNS (IXML_Document* doc, DOMString names- paceURI, DOMString localName) <i>Returns a NodeList of Elements that match the given local name and namespace URI in the order they are encountered in a preorder traversal of the Document tree.</i>	30
4.4.18	IXML_Element*	ixmlDocument_getElementById (IXML_Document* doc, DOMString tagName) <i>Returns the Element whose ID matches that given id.</i>	31
4.4.19	void	ixmlDocument_free (IXML_Document* doc) <i>Frees a Document object and all Nodes associated with it.</i>	31
4.4.20	int	ixmlDocument_importNode (IXML_Document* doc, IXML_Node* importNode, BOOL deep, IXML_Node** rtNode)	

*Imports a **Node** from another **Document** into this **Document**.* 32

The **Document** interface represents the entire XML document. In essence, it is the root of the document tree and provides the primary interface to the elements of the document. For more information, refer to the *Interface Document* section in the DOM2Core.

4.4.1

```
void ixmlDocument_init (IXML_Document* nodeptr )
```

*Initializes a **Document** node.*

Initializes a **Document** node.

Return Value: [void] This function does not return a value.
Parameters: nodeptr The **Document** node to initialize.

4.4.2

```
int ixmlDocument_createDocumentEx (IXML_Document** doc )
```

*Creates a new empty **Document** node.*

Creates a new empty **Document** node. The **ixmlDocument_createDocumentEx** API differs from the **ixmlDocument_createDocument** API in that it returns an error code describing the reason for the failure rather than just NULL.

Return Value: [int] An integer representing one of the following:

- **IXML_SUCCESS:** The operation completed successfully.
- **IXML_INSUFFICIENT_MEMORY:** Not enough free memory exists to complete this operation.

Parameters: doc Pointer to a **Document** where the new object will be stored.

4.4.3

```
IXML_Document* ixmlDocument_createDocument ()
```

*Creates a new empty **Document** node.*

Creates a new empty **Document** node.

Return Value: [Document*] A pointer to the new **Document** or NULL on failure.

4.4.4

```
int ixmlDocument_createElementEx (IXML_Document*      doc,
                                   DOMString             tagName,
                                   IXML_Element**        rtElement )
```

*Creates a new **Element** node with the given tag name.*

Creates a new **Element** node with the given tag name. The new **Element** node has a **nodeName** of **tagName** and the **localName**, **prefix**, and **namespaceURI** set to NULL. To create an **Element** with a namespace, see **ixmlDocument_createElementNS**.

The **ixmlDocument_createElementEx** API differs from the **ixmlDocument_createElement** API in that it returns an error code describing the reason for failure rather than just NULL.

Return Value: [int] An integer representing one of the following:

- **IXML_SUCCESS**: The operation completed successfully.
- **IXML_INVALID_PARAMETER**: Either **doc** or **tagName** is NULL.
- **IXML_INSUFFICIENT_MEMORY**: Not enough free memory exists to complete this operation.

Parameters:

doc	The owner Document of the new node.
tagName	The tag name of the new Element node.
rtElement	Pointer to an Element where the new object will be stored.

4.4.5

```

XML_Element* ixmlDocument_createElement (XML_Document*
                                           doc,  DOMString tag-
                                           Name )

```

*Creates a new **Element** node with the given tag name.*

Creates a new **Element** node with the given tag name. The new **Element** node has a **nodeName** of **tagName** and the **localName**, **prefix**, and **namespaceURI** set to NULL. To create an **Element** with a namespace, see **ixmlDocument_createElementNS**.

Return Value: [Document*] A pointer to the new **Element** or NULL on failure.
Parameters: doc The owner **Document** of the new node.
 tagName The tag name of the new **Element** node.

4.4.6

```

int ixmlDocument_createTextNodeEx (XML_Document* doc,  DOM-
                                     String data,  XML_Node**
                                     textNode )

```

*Creates a new **Text** node with the given data.*

Creates a new **Text** node with the given data. The **ixmlDocument_createTextNodeEx** API differs from the **ixmlDocument_createTextNode** API in that it returns an error code describing the reason for failure rather than just NULL.

Return Value: [int] An integer representing one of the following:

- **XML_SUCCESS**: The operation completed successfully.
- **XML_INVALID_PARAMETER**: Either **doc** or **data** is NULL.
- **XML_INSUFFICIENT_MEMORY**: Not enough free memory exists to complete this operation.

Parameters: doc The owner **Document** of the new node.
 data The data to associate with the new **Text** node.
 textNode A pointer to a **Node** where the new object will be stored.

4.4.7

```

XML_Node* xmlDoc_createTextNode (XML_Document* doc,
                                   DOMString data )

```

*Creates a new **Text** node with the given data.*

Creates a new **Text** node with the given data.

Return Value: [Node*] A pointer to the new **Node** or NULL on failure.
Parameters: doc The owner **Document** of the new node.
data The data to associate with the new **Text** node.

4.4.8

```

int xmlDoc_createCDATASectionEx (XML_Document*
                                   doc, DOMString data,
                                   XML_CDATASection**
                                   cdNode )

```

*Creates a new **CDATASection** node with given data.*

Creates a new **CDATASection** node with given data.

The `xmlDoc_createCDATASectionEx` API differs from the `xmlDoc_createCDATASection` API in that it returns an error code describing the reason for failure rather than just NULL.

Return Value: [int] An integer representing one of the following:

- **XML_SUCCESS:** The operation completed successfully.
- **XML_INVALID_PARAMETER:** Either **doc** or **data** is NULL.
- **XML_INSUFFICIENT_MEMORY:** Not enough free memory exists to complete this operation.

Parameters:

doc The owner **Document** of the new node.

data The data to associate with the new **CDATASection** node.

cdNode A pointer to a **Node** where the new object will be stored.

4.4.9

```
IXML_CDATASection*      ixmlDocument_createCDATASection
(IXML_Document* doc, DOMString data )
```

*Creates a new **CDATASection** node with given data.*

Creates a new **CDATASection** node with given data.

Return Value: [CDATASection*] A pointer to the new **CDATASection** or NULL on failure.

Parameters: **doc** The owner **Document** of the new node.
data The data to associate with the new **CDATA-Section** node.

4.4.10

```
IXML_Attr* ixmlDocument_createAttribute (IXML_Document* doc,
                                           char* name )
```

*Creates a new **Attr** node with the given name.*

Creates a new **Attr** node with the given name.

Return Value: [Attr*] A pointer to the new **Attr** or NULL on failure.

Parameters: **doc** The owner **Document** of the new node.
name The name of the new attribute.

4.4.11

```
int ixmlDocument_createAttributeEx (IXML_Document* doc, char*
                                     name, IXML_Attr** attrNode )
```

*Creates a new **Attr** node with the given name.*

Creates a new **Attr** node with the given name.

The **ixmlDocument_createAttributeEx** API differs from the **ixmlDocument_createAttribute** API in that it returns an error code describing the reason for failure rather than just NULL.

Return Value: [int] An integer representing one of the following:

- **IXML_SUCCESS:** The operation completed successfully.
- **IXML_INVALID_PARAMETER:** Either **doc** or **name** is NULL.
- **IXML_INSUFFICIENT_MEMORY:** Not enough free memory exists to complete this operation.

Parameters:

doc	The owner Document of the new node.
name	The name of the new attribute.
attrNode	A pointer to a Attr where the new object will be stored.

4.4.12

```
IXML_NodeList*      ixmlDocument_getElementsByTagName
(IXML_Document* doc, DOMString tagName )
```

*Returns a **NodeList** of all **Elements** that match the given tag name in the order in which they were encountered in a preorder traversal of the **Document** tree.*

Returns a **NodeList** of all **Elements** that match the given tag name in the order in which they were encountered in a preorder traversal of the **Document** tree.

Return Value: [NodeList*] A pointer to a **NodeList** containing the matching items or NULL on an error.

Parameters:

doc	The Document to search.
tagName	The tag name to find.

4.4.13

```
int ixmlDocument_createElementNSEx (IXML_Document*      doc,
                                     DOMString      namespaceURI,
                                     DOMString      qualifiedName,
                                     IXML_Element** rtElement )
```

*Creates a new **Element** node in the given qualified name and namespace URI.*

Creates a new **Element** node in the given qualified name and namespace URI.

The **ixmlDocument_createElementNSEx** API differs from the **ixmlDocument_createElementNS** API in that it returns an error code describing the reason for failure rather than just NULL.

Return Value:	[int]	An integer representing one of the following: <ul style="list-style-type: none"> • IXML_SUCCESS: The operation completed successfully. • IXML_INVALID_PARAMETER: Either doc, namespaceURI, or qualifiedName is NULL. • IXML_INSUFFICIENT_MEMORY: Not enough free memory exists to complete this operation.
Parameters:	doc namespaceURI qualifiedName rtElement	The owner Document of the new node. The namespace URI for the new Element . The qualified name of the new Element . A pointer to an Element where the new object will be stored.

4.4.14

```
IXML_Element* ixmlDocument_createElementNS (IXML_Document*
doc, DOMString namespaceURI, DOMString qualifiedName )
```

*Creates a new **Element** node in the given qualified name and namespace URI.*

Creates a new **Element** node in the given qualified name and namespace URI.

Return Value:	[Element*]	A pointer to the new Element or NULL on failure.
Parameters:	doc namespaceURI qualifiedName	The owner Document of the new node. The namespace URI for the new Element . The qualified name of the new Element .

4.4.15

```
int ixmlDocument_createAttributeNSEx (IXML_Document* doc,
DOMString namespaceURI,
DOMString qualifiedName,
IXML_Attr** attrNode )
```

*Creates a new **Attr** node with the given qualified name and namespace URI.*

Creates a new **Attr** node with the given qualified name and namespace URI.

The **ixmlDocument_createAttributeNSEx** API differs from the **ixmlDocument_createAttributeNS** API in that it returns an error code describing the reason for failure rather than just NULL.

Return Value:	[int]	An integer representing one of the following: <ul style="list-style-type: none"> • IXML_SUCCESS: The operation completed successfully. • IXML_INVALID_PARAMETER: Either doc, namespaceURI, or qualifiedName is NULL. • IXML_INSUFFICIENT_MEMORY: Not enough free memory exists to complete this operation.
Parameters:	doc namespaceURI qualifiedName attrNode	The owner Document of the new Attr . The namespace URI for the attribute. The qualified name of the attribute. A pointer to an Attr where the new object will be stored.

4.4.16

```

IXML_Attr* ixmlDocument_createAttributeNS (IXML_Document*
                                           doc,          DOMString
                                           namespaceURI,
                                           DOMString      quali-
                                           fiedName )

```

*Creates a new **Attr** node with the given qualified name and namespace URI.*

Creates a new **Attr** node with the given qualified name and namespace URI.

Return Value:	[Attr*]	A pointer to the new Attr or NULL on failure.
Parameters:	doc namespaceURI qualifiedName	The owner Document of the new Attr . The namespace URI for the attribute. The qualified name of the attribute.

4.4.17

```

IXML_NodeList* ixmlDocument_getElementsByTagNameNS
(IXML_Document* doc, DOMString namespaceURI, DOMString lo-
calName )

```

*Returns a **NodeList** of **Elements** that match the given local name and namespace URI in the order they are encountered in a preorder traversal of the **Document** tree.*

Returns a **NodeList** of **Elements** that match the given local name and namespace URI in the

order they are encountered in a preorder traversal of the **Document** tree. Either **namespaceURI** or **localName** can be the special "*" character, which matches any namespace or any local name respectively.

Return Value:	[NodeList*]	A pointer to a NodeList containing the matching items or NULL on an error.
Parameters:	doc	The Document to search.
	namespaceURI	The namespace of the elements to find or "*" to match any namespace.
	localName	The local name of the elements to find or "*" to match any local name.

4.4.18

```
IXML_Element* ixmlDocument_getElementById (IXML_Document*
doc, DOMString tagName )
```

*Returns the **Element** whose ID matches that given id.*

Returns the **Element** whose ID matches that given id.

Return Value:	[Element*]	A pointer to the matching Element or NULL on an error.
Parameters:	doc	The owner Document of the Element .
	tagName	The name of the Element .

4.4.19

```
void ixmlDocument_free (IXML_Document* doc )
```

*Frees a **Document** object and all **Nodes** associated with it.*

Frees a **Document** object and all **Nodes** associated with it. Any **Nodes** extracted via any other interface function, e.g. **ixmlDocument_GetElementById**, become invalid after this call unless explicitly cloned.

Return Value:	[void]	This function does not return a value.
Parameters:	doc	The Document to free.

4.4.20

```
int ixmlDocument_importNode (IXML_Document* doc, IXML_Node*
                             importNode,          BOOL      deep,
                             IXML_Node** rtNode )
```

*Imports a **Node** from another **Document** into this **Document**.*

Imports a **Node** from another **Document** into this **Document**. The new **Node** does not have a parent node: it is a clone of the original **Node** with the **ownerDocument** set to **doc**. The **deep** parameter controls whether all the children of the **Node** are imported. Refer to the DOM2-Core recommendation for details on importing specific node types.

Return Value:

[int] An integer representing one of the following:

- **IXML_SUCCESS**: The operation completed successfully.
- **IXML_INVALID_PARAMETER**: Either **doc** or **importNode** is not a valid pointer.
- **IXML_NOT_SUPPORTED_ERR**: **importNode** is a **Document**, which cannot be imported.
- **IXML_FAILED**: The import operation failed because the **Node** to be imported could not be cloned.

Parameters:

doc	The Document into which to import.
importNode	The Node to import.
deep	TRUE to import all children of importNode or FALSE to import only the root node.
rtNode	A pointer to a new Node owned by doc .

4.5

Interface *Element***Names**

- | | | | | |
|-------|-----------------|--------------------------------------------------------------------------|---------------------------------------------------------------|----|
| 4.5.1 | void | ixmlElement_init (IXML_Element* element) | <i>Initializes a IXML_Element node. ...</i> | 35 |
| 4.5.2 | const DOMString | ixmlElement_getTagName (IXML_Element* element) | <i>Returns the name of the tag as a constant string.</i> | 35 |
| 4.5.3 | DOMString | ixmlElement_getAttribute (IXML_Element* element, DOMString name) | | |

			<i>Retrieves an attribute of an Element by name.</i>	35
4.5.4	int	ixmlElement_setAttribute (IXML_Element* element, DOMString name, DOMString value)	<i>Adds a new attribute to an Element. ..</i>	36
4.5.5	int	ixmlElement_removeAttribute (IXML_Element* element, DOMString name)	<i>Removes an attribute by name.</i>	36
4.5.6	IXML_Attr*	ixmlElement_getAttributeNode (IXML_Element* element, DOMString name)	<i>Retrieves an attribute node by name. ...</i>	37
4.5.7	int	ixmlElement_setAttributeNode (IXML_Element* element, IXML_Attr* newAttr, IXML_Attr** rtAttr)	<i>Adds a new attribute node to an Element.</i>	37
4.5.8	int	ixmlElement_removeAttributeNode (IXML_Element* element, IXML_Attr* oldAttr, IXML_Attr** rtAttr)	<i>Removes the specified attribute node from an Element.</i>	38
4.5.9	IXML_NodeList*	ixmlElement_getElementsByTagName (IXML_Element* element, DOMString tagName)	<i>Returns a NodeList of all descendant El- ements with a given tag name, in the order in which they are encountered in a pre-order traversal of this Element tree.</i>	38
4.5.10	DOMString	ixmlElement_getAttributeNS (IXML_Element* element, DOMString namespaceURI, DOMString localname)	<i>Retrieves an attribute value using the local name and namespace URI.</i>	39
4.5.11	int	ixmlElement_setAttributeNS (IXML_Element* element, DOMString namespaceURI, DOMString qualifiedName, DOMString value)	<i>Adds a new attribute to an Element using the local name and namespace URI.</i>	39
4.5.12	int	ixmlElement_removeAttributeNS (IXML_Element* element, DOMString namespaceURI, DOMString localName)		

			<i>Removes an attribute using the namespace URI and local name.</i>	40
4.5.13	IXML_Attr*	ixmlElement_getAttributeNodeNS (IXML_Element* element, DOMString namespaceURI, DOMString localName)	<i>Retrieves an Attr node by local name and namespace URI.</i>	40
4.5.14	int	ixmlElement_setAttributeNodeNS (IXML_Element* element, IXML_Attr* newAttr, IXML_Attr** rcAttr)	<i>Adds a new attribute node.</i>	41
4.5.15	IXML_NodeList*	ixmlElement_getElementsByTagNameNS (IXML_Element* element, DOMString namespaceURI, DOMString localName)	<i>Returns a NodeList of all descendant Elements with a given tag name, in the order in which they are encountered in the pre-order traversal of the Element tree.</i>	41
4.5.16	BOOL	ixmlElement_hasAttribute (IXML_Element* element, DOMString name)	<i>Queries whether the Element has an attribute with the given name or a default value.</i>	42
4.5.17	BOOL	ixmlElement_hasAttributeNS (IXML_Element* element, DOMString namespaceURI, DOMString localName)	<i>Queries whether the Element has an attribute with the given local name and namespace URI or has a default value for that attribute.</i>	42
4.5.18	void	ixmlElement_free (IXML_Element* element)	<i>Frees the given Element and any subtree of the Element.</i>	42

The **Element** interface represents an element in an XML document. Only **Elements** are allowed to have attributes, which are stored in the **attributes** member of a **Node**. The **Element** interface extends the **Node** interface and adds more operations to manipulate attributes.

4.5.1

```
void ixmlElement_init (IXML_Element* element )
```

*Initializes a **IXML_Element** node.*

Initializes a **IXML_Element** node.

Return Value: [void] This function does not return a value.
Parameters: element The **Element** to initialize.

4.5.2

```
const DOMString ixmlElement_getTagName (IXML_Element* element
                                         )
```

Returns the name of the tag as a constant string.

Returns the name of the tag as a constant string.

Return Value: [const DOMString] A **DOMString** representing the name of the **Element**.
Parameters: element The **Element** from which to retrieve the name.

4.5.3

```
DOMString ixmlElement_getAttribute (IXML_Element* element,
                                     DOMString name )
```

*Retrieves an attribute of an **Element** by name.*

Retrieves an attribute of an **Element** by name.

Return Value: [DOMString] A **DOMString** representing the value of the attribute.
Parameters: element The **Element** from which to retrieve the attribute.
 name The name of the attribute to retrieve.

4.5.4

```
int ixmlElement_setAttribute (IXML_Element* element, DOMString
                              name, DOMString value )
```

*Adds a new attribute to an **Element**.*

Adds a new attribute to an **Element**. If an attribute with the same name already exists, the attribute value will be updated with the new value in **value**.

Return Value: [int] An integer representing of the following:

- **IXML_SUCCESS**: The operation completed successfully.
- **IXML_INVALID_PARAMETER**: Either **element**, **name**, or **value** is NULL.
- **IXML_INVALID_CHARACTER_ERR**: **name** contains an illegal character.
- **IXML_INSUFFICIENT_MEMORY**: Not enough free memory exists to complete the operation.

Parameters:

element	The Element on which to set the attribute.
name	The name of the attribute.
value	The value of the attribute. Note that this is a non-parsed string and any markup must be escaped.

4.5.5

```
int ixmlElement_removeAttribute (IXML_Element* element, DOM-String name )
```

Removes an attribute by name.

Removes an attribute by name.

Return Value: [int] An integer representing one of the following:

- **IXML_SUCCESS**: The operation completed successfully.
- **IXML_INVALID_PARAMETER**: Either **element** or **name** is NULL.

Parameters:

element	The Element from which to remove the attribute.
name	The name of the attribute to remove.

4.5.6

```
IXML_Attr* ixmlElement_getAttributeNode (IXML_Element* element, DOMString name )
```

Retrieves an attribute node by name.

Retrieves an attribute node by name. See **ixmlElement_getAttributeNodeNS** to retrieve an attribute node using a qualified name or namespace URI.

Return Value: [Attr*] A pointer to the attribute matching **name** or NULL on an error.

Parameters:

element	The Element from which to get the attribute node.
name	The name of the attribute node to find.

4.5.7

```
int ixmlElement_setAttributeNode (IXML_Element*      element,
                                  IXML_Attr*          newAttr,
                                  IXML_Attr** rtAttr )
```

*Adds a new attribute node to an **Element**.*

Adds a new attribute node to an **Element**. If an attribute already exists with **newAttr** as a name, it will be replaced with the new one and the old one will be returned in **rtAttr**.

Return Value: [int] An integer representing one of the following:

- **IXML_SUCCESS**: The operation completed successfully.
- **IXML_INVALID_PARAMETER**: Either **element** or **newAttr** is NULL.
- **IXML_WRONG_DOCUMENT_ERR**: **newAttr** does not belong to the same one as **element**.
- **IXML_INUSE_ATTRIBUTE_ERR**: **newAttr** is already an attribute of another **Element**.

Parameters:

element	The Element in which to add the new attribute.
newAttr	The new Attr to add.
rtAttr	A pointer to an Attr where the old Attr will be stored. This will have a NULL if no prior node existed.

4.5.8

```
int ixmlElement_removeAttributeNode (IXML_Element*      ele-
                                     ment, IXML_Attr*    oldAttr,
                                     IXML_Attr** rtAttr )
```

*Removes the specified attribute node from an **Element**.*

Removes the specified attribute node from an **Element**.

Return Value: [int] An integer representing one of the following:

- **IXML_SUCCESS**: The operation completed successfully.
- **IXML_INVALID_PARAMETER**: Either **element** or **oldAttr** is NULL.
- **IXML_NOT_FOUND_ERR**: **oldAttr** is not among the list attributes of **element**.

Parameters:

element	The Element from which to remove the attribute.
oldAttr	The attribute to remove from the Element .
rtAttr	A pointer to an attribute in which to place the removed attribute.

4.5.9

IXML_NodeList* **ixmlElement_getElementsByTagName**
(IXML_Element* element, DOMString tagName)

*Returns a **NodeList** of all descendant **Elements** with a given tag name, in the order in which they are encountered in a pre-order traversal of this **Element** tree.*

Returns a **NodeList** of all *descendant* **Elements** with a given tag name, in the order in which they are encountered in a pre-order traversal of this **Element** tree.

Return Value: [NodeList*] A **NodeList** of the matching **Elements** or NULL on an error.

Parameters:

element	The Element from which to start the search.
tagName	The name of the tag for which to search.

4.5.10

DOMString **ixmlElement_getAttributeNS** (IXML_Element* element,
DOMString namespaceURI,
DOMString localname)

Retrieves an attribute value using the local name and namespace URI.

Retrieves an attribute value using the local name and namespace URI.

Return Value: [DOMString] A **DOMString** representing the value of the matching attribute.

Parameters:

element	The Element from which to get the attribute value.
namespaceURI	The namespace URI of the attribute.
localname	The local name of the attribute.

4.5.11

```
int ixmlElement_setAttributeNS (IXML_Element* element, DOM-
                                String namespaceURI, DOMString
                                qualifiedName, DOMString value )
```

*Adds a new attribute to an **Element** using the local name and namespace URI.*

Adds a new attribute to an **Element** using the local name and namespace URI. If another attribute matches the same local name and namespace, the prefix is changed to be the prefix part of the **qualifiedName** and the value is changed to **value**.

Return Value: [int] An integer representing one of the following:

- **IXML_SUCCESS**: The operation completed successfully.
- **IXML_INVALID_PARAMETER**: Either **element**, **namespaceURI**, **qualifiedName**, or **value** is NULL.
- **IXML_INVALID_CHARACTER_ERR**: **qualifiedName** contains an invalid character.
- **IXML_NAMESPACE_ERR**: Either the **qualifiedName** or **namespaceURI** is malformed. Refer to the DOM2-Core for possible reasons.
- **IXML_INSUFFICIENT_MEMORY**: Not enough free memory exist to complete the operation.
- **IXML_FAILED**: The operation could not be completed.

Parameters:

element	The Element on which to set the attribute.
namespaceURI	The namespace URI of the new attribute.
qualifiedName	The qualified name of the attribute.
value	The new value for the attribute.

4.5.12

```
int ixmlElement_removeAttributeNS (IXML_Element* element, DOM-
                                   String namespaceURI, DOM-
                                   String localName )
```

Removes an attribute using the namespace URI and local name.

Removes an attribute using the namespace URI and local name.

Return Value: [int] An integer representing one of the following:

- **IXML_SUCCESS**: The operation completed successfully.
- **IXML_INVALID_PARAMETER**: Either **element**, **namespaceURI**, or **localName** is NULL.

Parameters:

element	The Element from which to remove the attribute.
namespaceURI	The namespace URI of the attribute.
localName	The local name of the attribute.

4.5.13

```
IXML_Attr* ixmlElement_getAttributeNodeNS (IXML_Element* ele-
                                             ment, DOMString namespaceURI, DOMString localName )
```

*Retrieves an **Attr** node by local name and namespace URI.*

Retrieves an **Attr** node by local name and namespace URI.

Return Value: [Attr*] A pointer to an **Attr** or NULL on an error.

Parameters:

element	The Element from which to get the attribute.
namespaceURI	The namespace URI of the attribute.
localName	The local name of the attribute.

4.5.14

```
int ixmlElement_setAttributeNodeNS (IXML_Element* element,
                                       IXML_Attr* newAttr,
                                       IXML_Attr** rcAttr )
```

Adds a new attribute node.

Adds a new attribute node. If an attribute with the same local name and namespace URI already exists in the **Element**, the existing attribute node is replaced with **newAttr** and the old returned in **rcAttr**.

Return Value: [int] An integer representing one of the following:

- **IXML_SUCCESS**: The operation completed successfully.
- **IXML_INVALID_PARAMETER**: Either **element** or **newAttr** is NULL.
- **IXML_WRONG_DOCUMENT_ERR**: **newAttr** does not belong to the same document as **element**.
- **IXML_INUSE_ATTRIBUTE_ERR**: **newAttr** already is an attribute of another **Element**.

Parameters:

element	The Element in which to add the attribute node.
newAttr	The new Attr to add.
rcAttr	A pointer to the replaced Attr , if it exists.

4.5.15

```
IXML_NodeList*      ixmlElement_getElementsByTagNameNS
(IXML_Element* element, DOMString namespaceURI, DOMString
localName )
```

*Returns a **NodeList** of all descendant **Elements** with a given tag name, in the order in which they are encountered in the pre-order traversal of the **Element** tree.*

Returns a **NodeList** of all *descendant* **Elements** with a given tag name, in the order in which they are encountered in the pre-order traversal of the **Element** tree.

Return Value: [NodeList*] A **NodeList** of matching **Elements** or NULL on an error.

Parameters:

element	The Element from which to start the search.
namespaceURI	The namespace URI of the Elements to find.
localName	The local name of the Elements to find.

4.5.16

```
BOOL ixmlElement_hasAttribute (IXML_Element* element, DOM-
String name )
```

*Queries whether the **Element** has an attribute with the given name or a default value.*

Queries whether the **Element** has an attribute with the given name or a default value.

Return Value: [BOOL] TRUE if the **Element** has an attribute with this name or has a default value for that attribute, otherwise FALSE.

Parameters:

element	The Element on which to check for an attribute.
name	The name of the attribute for which to check.

4.5.17

```

BOOL ixmlElement_hasAttributeNS (IXML_Element* element, DOM-
                                String namespaceURI, DOM-
                                String localName )

```

*Queries whether the **Element** has an attribute with the given local name and namespace URI or has a default value for that attribute.*

Queries whether the **Element** has an attribute with the given local name and namespace URI or has a default value for that attribute.

Return Value: [BOOL] TRUE if the **Element** has an attribute with the given namespace and local name or has a default value for that attribute, otherwise FALSE.

Parameters:

element	The Element on which to check for the attribute.
namespaceURI	The namespace URI of the attribute.
localName	The local name of the attribute.

4.5.18

```

void ixmlElement_free (IXML_Element* element )

```

*Frees the given **Element** and any subtree of the **Element**.*

Frees the given **Element** and any subtree of the **Element**.

Return Value: [void] This function does not return a value.

Parameters:

element	The Element to free.
----------------	-----------------------------

4.6

Interface *NamedNodeMap***Names**

- 4.6.1 unsigned long **ixmlNamedNodeMap_getLength** (IXML_NamedNodeMap*
nnMap)
Returns the number of items contained in this NamedNodeMap. 44
- 4.6.2 IXML_Node* **ixmlNamedNodeMap_getNamedItem**
(IXML_NamedNodeMap*
nnMap,
DOMString name)
Retrieves a Node from the NamedNodeMap by name. 44
- 4.6.3 IXML_Node* **ixmlNamedNodeMap_setNamedItem**
(IXML_NamedNodeMap*
nnMap,
IXML_Node* arg)
Adds a new Node to the NamedNodeMap using the Node name attribute. 45
- 4.6.4 IXML_Node* **ixmlNamedNodeMap_removeNamedItem**
(IXML_NamedNodeMap*
nnMap,
DOMString
name)
Removes a Node from a NamedNodeMap specified by name. . 45
- 4.6.5 IXML_Node* **ixmlNamedNodeMap_item** (IXML_NamedNodeMap* nnMap,
unsigned long index)
Retrieves a Node from a NamedNodeMap specified by a numerical index. 46
- 4.6.6 IXML_Node* **ixmlNamedNodeMap_getNamedItemNS**
(IXML_NamedNodeMap*
nnMap,
DOMString*
namespaceURI,
DOMString
localName)
Retrieves a Node from a NamedNodeMap specified by namespace URI and local name. 46
- 4.6.7 IXML_Node* **ixmlNamedNodeMap_setNamedItemNS**
(IXML_NamedNodeMap*
nnMap,
IXML_Node* arg
)

		<i>Adds a new Node to the NamedNodeMap using the Node local name and namespace URI attributes.</i>	
		46
4.6.8	IXML_Node* ixmlNamedNodeMap_removeNamedItemNS	(IXML_NamedNodeMap* nnMap, DOMString namespaceURI, DOMString localName)	
		<i>Removes a Node from a NamedNodeMap specified by namespace URI and local name.</i>	47
4.6.9	void ixmlNamedNodeMap_free	(IXML_NamedNodeMap* nnMap)	
		<i>Frees a NamedNodeMap.</i>	47

A **NamedNodeMap** object represents a list of objects that can be accessed by name. A **NamedNodeMap** maintains the objects in no particular order. The **Node** interface uses a **NamedNodeMap** to maintain the attributes of a node.

4.6.1

unsigned long **ixmlNamedNodeMap_getLength** (IXML_NamedNodeMap* nnMap)

*Returns the number of items contained in this **NamedNodeMap**.*

Returns the number of items contained in this **NamedNodeMap**.

Return Value: [unsigned long] The number of nodes in this map.
Parameters: nnMap The **NamedNodeMap** from which to retrieve the size.

4.6.2

IXML_Node* **ixmlNamedNodeMap_getNamedItem**
 (IXML_NamedNodeMap* nnMap, DOMString name)

*Retrieves a **Node** from the **NamedNodeMap** by name.*

Retrieves a **Node** from the **NamedNodeMap** by name.

Return Value: [Node*] A **Node** or NULL if there is an error.
Parameters: nnMap The **NamedNodeMap** to search.
name The name of the **Node** to find.

4.6.3

IXML_Node* **ixmlNamedNodeMap_setNamedItem**
(IXML_NamedNodeMap* nnMap, IXML_Node* arg)

*Adds a new **Node** to the **NamedNodeMap** using the **Node** name attribute.*

Adds a new **Node** to the **NamedNodeMap** using the **Node** name attribute.

Return Value: [Node*] The old **Node** if the new **Node** replaces it or NULL if the **Node** was not in the **NamedNodeMap** before.
Parameters: nnMap The **NamedNodeMap** in which to add the new **Node**.
arg The new **Node** to add to the **NamedNodeMap**.

4.6.4

IXML_Node* **ixmlNamedNodeMap_removeNamedItem**
(IXML_NamedNodeMap* nnMap, DOMString name)

*Removes a **Node** from a **NamedNodeMap** specified by name.*

Removes a **Node** from a **NamedNodeMap** specified by name.

Return Value: [Node*] A pointer to the **Node**, if found, or NULL if it wasn't.
Parameters: nnMap The **NamedNodeMap** from which to remove the item.
name The name of the item to remove.

4.6.5

IXML_Node* **ixmlNamedNodeMap_item** (IXML_NamedNodeMap* nn-
Map, unsigned long index)

*Retrieves a **Node** from a **NamedNodeMap** specified by a numerical index.*

Retrieves a **Node** from a **NamedNodeMap** specified by a numerical index.

Return Value: [Node*] A pointer to the **Node**, if found, or NULL if it wasn't.

Parameters:

nnMap	The NamedNodeMap from which to remove the Node .
index	The index into the map to remove.

4.6.6

```

IXML_Node*      ixmlNamedNodeMap_getNamedItemNS
(IXML_NamedNodeMap* nnMap, DOMString* namespaceURI, DOM-
String localName )
  
```

*Retrieves a **Node** from a **NamedNodeMap** specified by namespace URI and local name.*

Retrieves a **Node** from a **NamedNodeMap** specified by namespace URI and local name.

Return Value: [Node*] A pointer to the **Node**, if found, or NULL if it wasn't

Parameters:

nnMap	The NamedNodeMap from which to remove the Node .
namespaceURI	The namespace URI of the Node to remove.
localName	The local name of the Node to remove.

4.6.7

```

IXML_Node*      ixmlNamedNodeMap_setNamedItemNS
(IXML_NamedNodeMap* nnMap, IXML_Node* arg )
  
```

*Adds a new **Node** to the **NamedNodeMap** using the **Node** local name and namespace URI attributes.*

Adds a new **Node** to the **NamedNodeMap** using the **Node** local name and namespace URI attributes.

Return Value: [Node*] The old **Node** if the new **Node** replaces it or NULL if the **Node** was not in the **NamedNodeMap** before.

Parameters:

nnMap	The NamedNodeMap in which to add the Node .
arg	The Node to add to the map.

4.6.8

```

IXML_Node*      ixmlNamedNodeMap_removeNamedItemNS
(IXML_NamedNodeMap* nnMap,  DOMString namespaceURI,  DOM-
String localName )

```

*Removes a **Node** from a **NamedNodeMap** specified by namespace URI and local name.*

Removes a **Node** from a **NamedNodeMap** specified by namespace URI and local name.

Return Value: [Node*] A pointer to the **Node**, if found, or NULL if it wasn't.

Parameters: nnMap The **NamedNodeMap** from which to remove the **Node**.

namespaceURI The namespace URI of the **Node** to remove.

localName The local name of the **Node** to remove.

4.6.9

```

void ixmlNamedNodeMap_free (IXML_NamedNodeMap* nnMap )

```

*Frees a **NamedNodeMap**.*

Frees a **NamedNodeMap**. The **Nodes** inside the map are not freed, just the **NamedNodeMap** object.

Return Value: [void] This function does not return a value.

Parameters: nnMap The **NamedNodeMap** to free.

4.7

Interface *NodeList*

Names

- 4.7.1 IXML_Node* **ixmlNodeList_item** (IXML_NodeList* nList, unsigned long index)
*Retrieves a **Node** from a **NodeList** specified by a numerical index. 48*
- 4.7.2 unsigned long **ixmlNodeList_length** (IXML_NodeList* nList)
*Returns the number of **Nodes** in a **NodeList**. 48*
- 4.7.3 void **ixmlNodeList_free** (IXML_NodeList* nList)
*Frees a **NodeList** object. 48*

The **NodeList** interface abstracts an ordered collection of nodes. Note that changes to the underlying nodes will change the nodes contained in a **NodeList**. The DOM2-Core refers to this as being *live*.

4.7.1

```
IXML_Node* ixmlNodeList_item (IXML_NodeList* nList, unsigned long
                                index )
```

*Retrieves a **Node** from a **NodeList** specified by a numerical index.*

Retrieves a **Node** from a **NodeList** specified by a numerical index.

Return Value: [Node*] A pointer to a **Node** or NULL if there was an error.

Parameters: nList The **NodeList** from which to retrieve the **Node**.
index The index into the **NodeList** to retrieve.

4.7.2

```
unsigned long ixmlNodeList_length (IXML_NodeList* nList )
```

*Returns the number of **Nodes** in a **NodeList**.*

Returns the number of **Nodes** in a **NodeList**.

Return Value: [unsigned long] The number of **Nodes** in the **NodeList**.

Parameters: nList The **NodeList** for which to retrieve the number of **Nodes**.

4.7.3

```
void ixmlNodeList_free (IXML_NodeList* nList )
```

*Frees a **NodeList** object.*

Frees a **NodeList** object. Since the underlying **Nodes** are references, they are not freed using this operating. This only frees the **NodeList** object.

Return Value: [void] This function does not return a value.

Parameters: nList The **NodeList** to free.

IXML API

Names

5.1	DOMString	ixmlPrintNode (IXML_Node* doc)	<i>Renders a Node and all sub-elements into an XML text representation.</i>	49
5.2	DOMString	ixmlNodetoString (IXML_Node* doc)	<i>Renders a Node and all sub-elements into an XML text representation.</i>	50
5.3	IXML_Document*	ixmlParseBuffer (char* buffer)	<i>Parses an XML text buffer converting it into an IXML DOM representation. ...</i>	50
5.4	int	ixmlParseBufferEx (char* buffer, IXML_Document** doc)	<i>Parses an XML text buffer converting it into an IXML DOM representation. ...</i>	51
5.5	IXML_Document*	ixmlLoadDocument (char* xmlFile)	<i>Parses an XML text file converting it into an IXML DOM representation.</i>	51
5.6	int	ixmlLoadDocumentEx (char* xmlFile, IXML_Document** doc)	<i>Parses an XML text file converting it into an IXML DOM representation.</i>	52
5.7	DOMString	ixmlCloneDOMString (const DOMString src)	<i>Clones an existing DOMString.</i>	52
5.8	void	ixmlFreeDOMString (DOMString buf)	<i>Frees a DOMString.</i>	52

The IXML API contains utility functions that are not part of the standard DOM interfaces. They include functions to create a DOM structure from a file or buffer, create an XML file from a DOM structure, and manipulate DOMString objects.

5.1

DOMString **ixmlPrintNode** (IXML_Node* doc)

*Renders a **Node** and all sub-elements into an XML text representation.*

Renders a **Node** and all sub-elements into an XML text representation. The caller is required to free the **DOMString** returned from this function using **ixmlFreeDOMString** when it is no longer required.

Note that this function can be used for any **Node**-derived interface. A similar **ixmlPrintDocument** function is defined to avoid casting when printing whole documents. This function introduces lots of white space to print the **DOMString** in readable format.

Return Value: [DOMString] A **DOMString** with the XML text representation of the DOM tree or NULL on an error.

Parameters: doc The root of the **Node** tree to render to XML text.

5.2

DOMString ixmlNodetoString (IXML_Node* doc)

*Renders a **Node** and all sub-elements into an XML text representation.*

Renders a **Node** and all sub-elements into an XML text representation. The caller is required to free the **DOMString** returned from this function using **ixmlFreeDOMString** when it is no longer required.

Note that this function can be used for any **Node**-derived interface. A similar **ixmlPrintDocument** function is defined to avoid casting when printing whole documents.

Return Value: [DOMString] A **DOMString** with the XML text representation of the DOM tree or NULL on an error.

Parameters: doc The root of the **Node** tree to render to XML text.

5.3

IXML_Document* ixmlParseBuffer (char* buffer)

Parses an XML text buffer converting it into an IXML DOM representation.

Parses an XML text buffer converting it into an IXML DOM representation.

Return Value: [Document*] A **Document** if the buffer correctly parses or NULL on an error.

Parameters: buffer The buffer that contains the XML text to convert to a **Document**.

5.4

```
int ixmlParseBufferEx (char* buffer, IXML_Document** doc )
```

Parses an XML text buffer converting it into an IXML DOM representation.

Parses an XML text buffer converting it into an IXML DOM representation.

The **ixmlParseBufferEx** API differs from the **ixmlParseBuffer** API in that it returns an error code representing the actual failure rather than just **NULL**.

Return Value: [int] An integer representing one of the following:

- **IXML_SUCCESS:** The operation completed successfully.
- **IXML_INVALID_PARAMETER:** The **buffer** is not a valid pointer.
- **IXML_INSUFFICIENT_MEMORY:** Not enough free memory exists to complete this operation.

Parameters:

buffer The buffer that contains the XML text to convert to a **Document**.

doc A point to store the **Document** if file correctly parses or **NULL** on an error.

5.5

```
IXML_Document* ixmlLoadDocument (char* xmlFile )
```

Parses an XML text file converting it into an IXML DOM representation.

Parses an XML text file converting it into an IXML DOM representation.

Return Value: [Document*] A **Document** if the file correctly parses or **NULL** on an error.

Parameters: **xmlFile** The filename of the XML text to convert to a **Document**.

5.6

```
int ixmlLoadDocumentEx (char* xmlFile, IXML_Document** doc )
```

Parses an XML text file converting it into an IXML DOM representation.

Parses an XML text file converting it into an IXML DOM representation.

The **ixmlLoadDocumentEx** API differs from the **ixmlLoadDocument** API in that it returns a an error code representing the actual failure rather than just **NULL**.

Return Value: [int] An integer representing one of the following:

- **IXML_SUCCESS**: The operation completed successfully.
- **IXML_INVALID_PARAMETER**: The **xmlFile** is not a valid pointer.
- **IXML_INSUFFICIENT_MEMORY**: Not enough free memory exists to complete this operation.

Parameters:

xmlFile	The filename of the XML text to convert to a Document .
doc	A pointer to the Document if file correctly parses or NULL on an error.

5.7

DOMString ixmlCloneDOMString (const **DOMString** src)

*Clones an existing **DOMString**.*

Clones an existing **DOMString**.

Return Value: [**DOMString**] A new **DOMString** that is a duplicate of the original or **NULL** if the operation could not be completed.

Parameters: **src** The source **DOMString** to clone.

5.8

void ixmlFreeDOMString (**DOMString** buf)

*Frees a **DOMString**.*

Frees a **DOMString**.

Return Value: [**void**] This function does not return a value.

Parameters: **buf** The **DOMString** to free.